UNIVERSITÄT ZU LÜBECK
INSTITUTE OF MATHEMATICS AND
IMAGE COMPUTING

Diskrete und kontinuierliche Graph-Cut-Verfahren in der Bildverarbeitung

Discrete and Continuous Graph Cut Methods in Image Processing

Masterarbeit

im Rahmen des Studiengangs
Mathematik in Medizin und Lebenswissenschaften
der Universität zu Lübeck

Vorgelegt von
Annika Reinke, B.Sc.

Ausgegeben und betreut von
Prof. Dr. Jan Lellmann
Institute of Mathematics and Image Computing

Lübeck, den 18.11.2016

Eidesstattliche Erklärung

Ich versichere an Eides statt, die vorliegende Arbeit selbstständig und nur unter Benutzung der angegebenen Quellen und Hilfsmittel angefertigt zu haben.

Lübeck,
18.11.2016

Annika Reinke

Kurzfassung

Graph-Cut-Verfahren werden heutzutage erfolgreich in der Bildverarbeitung, speziell der Segmentierung, angewandt. Klassische diskrete Methoden bieten den Vorteil einer geringen Laufzeit, weisen jedoch häufig Diskretisierungsartefakte auf. In den letzten Jahren wurden spezielle kontinuierliche Graph-Cut-Verfahren entwickelt, welche diese Artefakte umgehen können. Der Nachteil solcher Algorithmen liegt darin, dass sie die Problemstruktur diskreter Graph-Cut-Methoden noch nicht optimal ausnutzen.
In dieser Arbeit sollen diskrete und kontinuierliche Graph-Cut-Verfahren gegenübergestellt werden. Ziel ist es zudem, die in einer Vorarbeit entwickelte *Methode des erweiterten Residualgraphen* hinsichtlich ihrer Laufzeit und Ergebnisse zu verbessern. Dazu werden insbesondere die Pfadsuche und die Kapazitätsbedingung für die Quelle und die Senke des Graphen untersucht. Zudem wird ein Ansatz vorgestellt, durch den mithilfe von *Quasizyklen* die Qualität des Verfahrens verbessert werden kann.
Abschließend lässt sich experimentell erkennen, dass die Methode Diskretisierungsartefakte umgehen kann. Zusätzlich erzeugt sie gute Segmentierungsergebnisse und benötigt weniger Iterationen zur Lösung des Problems als die diskreten Algorithmen, die zum Vergleich verwendet wurden.

Abstract

Graph Cut methods are successfully used in image processing, in particular for segmentation algorithms. Classical discrete methods offer the advantage of a shorter runtime but often exhibit discretization artifacts. In the last few years, specialized continuous Graph Cut techniques have been developed that can circumvent these artifacts. However, the disadvantage of such algorithms lies in the fact that they do not optimally exploit the problem structure of discrete Graph Cut methods.
In this thesis, discrete and continuous Graph Cut methods are compared. The goal is also to improve the *method of extended residual graph*, developed in a prior work, with regard to its runtime and quality. For this purpose the pathfinding and the capacity condition for the source and the sink of the graph are examined. In addition, an approach is presented which can improve the quality of the method by using *quasi-cycles*.
Finally, it can be seen experimentally that the method can circumvent discretization artifacts. Furthermore, it generates good segmentation results and requires less iterations solving the problem than the discrete algorithms used for comparison.

Danksagung

Mein besonderer Dank gilt meinem Betreuer Jan Lellmann. Danke, dass du dir immer die Zeit für mich genommen hast, mich bei meinen Problemen unterstützt und stets eine Antwort auf meine Fragen hattest. Die Arbeit an dem Thema hat mir sehr viel Spaß gemacht.

Vielen Dank auch an Hanns Teichert, dass Sie sich als Zweitprüfer der Arbeit zur Verfügung gestellt haben.

Ohne meine Familie wäre ich niemals bis hierhin gekommen, daher möchte ich euch an dieser Stelle meinen Dank ausrichten! Danke, dass ihr mich immer unterstützt und an mich glaubt. Vielen Dank auch für das Korrekturlesen.

Schließlich danke ich meinen Freunden für euren Beistand, eure Zustimmung und dass ihr mich in den guten und schwierigen Phasen immer zum Lachen bringen konntet.

Inhaltsverzeichnis

Kapitel 1: Einleitung

Bildsegmentierung stellt ein weites Anwendungsgebiet im Bereich der Bildverarbeitung dar. Dabei werden ähnliche Bildbereiche so zu einer Menge zusammengefasst, dass eine Partition des Bildes entsteht (siehe Abbildung 1). Dies dient insbesondere dazu, das Bild in eine geeignetere Repräsentation zu überführen, welche eine Analyse der wesentlichen Aspekte des Bildes vereinfachen kann. Anwendung finden Segmentierungen insbesondere in der medizinischen Bildverarbeitung. Hier unterstützen sie beispielsweise Klassifizierungen in der Computer- oder Magnetresonanztomographie. Darüber hinaus werden sie auch in der Objekterkennung oder im Bereich des maschinellen Sehens verwendet [28].

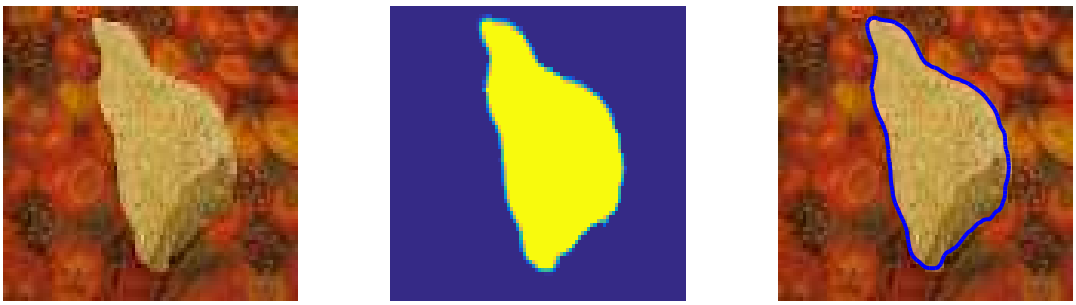


Abbildung 1. Segmentierung des im Originalbild (links) dargestellten Schwamms. Die beiden berechneten Mengen sind in der Mitte zu sehen. Das rechte Bild zeigt in blau die Kontur der Segmentierung im Originalbild.

Als Bild wird dabei eine Funktion $B : \mathbb{R}^d \to \mathbb{R}$ bezeichnet, welche einem Bildpunkt aus der Menge von Bildpunkten (*Pixeln*) $\mathcal{P}$ einen Intensitätswert zuordnet. Eine Segmentierung teilt die einzelnen Pixel abhängig von ihren Eigenschaften in verschiedene Mengen ein, wobei ähnliche Pixel demselben Segment angehören sollen. Zudem werden weitere globale Regularitätseigenschaften gefordert, beispielsweise sollten die Segmente glatte Ränder besitzen. Das Bild kann je nach Algorithmus in beliebig viele Mengen eingeteilt werden. Die in dieser Arbeit vorgestellten Methoden unterteilen das Bild in zwei Segmente. Die Zuordnung zu den einzelnen Mengen kann auf Basis unterschiedlicher Kriterien erfolgen. Beispiele sind die Intensität einzelner Bildpunkte oder die Textur des Bildes. Dazu existiert eine Vielzahl von automatisierten Lösungsansätzen [28]:

- *Kantenorientierte Verfahren* versuchen gezielt die Konturen eines Objektes zu ermitteln und somit eine Segmentierung zu erzielen [15, 52].
- Die einfachsten Verfahren liefern eine Zuordnung zu den verschiedenen Segmenten auf Basis eines *Schwellwerts* der Pixelintensitäten, so dass alle Bildpunkte, die ihn über-/ oder unterschreiten, in die zugehörige Menge eingeteilt werden [20, 45].
- Bei *regionenbasierten Verfahren* wie dem *Region-Growing* oder dem *Watershed-Algorithmus* wird mit der Annahme gearbeitet, dass benachbarte Bildregionen

ähnliche Eigenschaften besitzen. Sie vergleichen dazu einzelne Pixel mit deren Nachbarn [45, 52].

Eine weitere Klasse von Lösungsverfahren stellen sogenannte Graph-Cut-Methoden dar. Hier wird das Bild als Graph interpretiert und in zwei Segmente unterteilt, so dass die Summe der Gewichte auf den durch die Segmentgrenze geschnittenen Kanten minimiert wird. Bei der Verwendung von Graph-Cut-Verfahren zur Lösung eines Segmentierungsproblems werden die einzelnen Bildpunkte als Knoten eines Graphen dargestellt. Benachbarte Pixel werden durch Kanten verbunden und deren Kosten abhängig von der Anwendung bestimmt (siehe beispielsweise [9]). Das Ergebnis des Graph-Cut-Problems liefert die Aufteilung des Bildes in zwei Teilmengen und entspricht einer binären Segmentierung.

Üblicherweise werden Graph-Cut-Probleme als kombinatorische Energieminimierungsprobleme formuliert [33]. Bei vielen Anwendungen wird von einem regionen- und einem kantenbasierten Term ausgegangen [9]. Die Energie E ist dabei über

$$E(A) = \sum_{i \in \mathcal{P}} B_i(A_i) + \sum_{(v,w) \in E} R_{v,w}(A_v, A_w), \tag{1}$$

angegeben, wobei $A = (A_1, ..., A_p, ..., A_{|\mathcal{P}|}) \in \{0, 1\}$ die Zuordnung jedes Pixels zu dem jeweiligen Segment beschreibt. Die kantenbasierten Terme $R_{v,w}(A_v, A_w)$ sorgen dafür, dass ähnliche Pixel demselben Segment zugeordnet werden. Dagegen bevorzugen die regionenorientierten Terme $B_i(A_i)$ eine Zuordnung zu bestimmten Segmenten. Weiterhin lassen sich aus den Pixelintensitäten Histogramme auf Basis von Saatpunkten für den Vorder- und Hintergrund des Bildes erstellen, die zur Berechnung geeigneter Kosten B_i verwendet werden können.

Zur Lösung von Graph-Cut-Problemen wird in der Regel die duale Formulierung betrachtet und der Fluss auf den Kanten maximiert. Es existieren sehr schnelle Verfahren für die Optimierung dieser Probleme, die beispielsweise eine Segmentierung von Bildern einer Größe von 600×450 Bildpunkten in nur 5 Millisekunden lösen können [50]. Jedoch hat die diskrete Darstellung als Graph Nachteile: Da der Graph die Pixel des Bildes repräsentiert wird, ist der Aufbau der Knoten und Kanten durch das Eingabebild fest vorgegeben, wodurch die Schnittkanten nur entlang der Pixelgrenzen verlaufen können. Die Verbindung eines Knotens zu vier seiner Nachbarn birgt zudem eine gewisse Richtungsabhängigkeit. Dies kann dazu führen, dass die berechnete Segmentierung achsenparallele Konturen bevorzugt und Diskretisierungsartefakte aufweist [53]. Um diesem Problem entgegenzuwirken, wurden in den letzten Jahren verschiedene Lösungen vorgestellt, beispielsweise die Vergrößerung des Nachbarschaftssystems durch das Hinzufügen weiterer Kanten [10].

Eine andere Möglichkeit stellt die Übertragung des Graph-Cut-Problems in einen (orts-) kontinuierlichen Raum dar. Bereits 1983 wurde der erste Ansatz von Strang [49] vorgestellt. In den weiterführenden Jahren entwickelten beispielsweise Appleton und Talbot

[3] erfolgreiche kontinuierliche Methoden, welche zusätzlich in der Lage sind, dreidimensionale Strukturen zu segmentieren. Der Nachteil dieser kontinuierlichen Graph-Cut-Verfahren liegt darin, dass die Lösungsmethoden des Optimierungsproblems aufgrund der alternativen Repräsentation nicht direkt aus dem diskreten Fall übertragen werden können. Viele kontinuierliche Algorithmen sind generisch und nutzen die Struktur des Graph-Cut-Problems noch nicht optimal aus.

1.1 Aufbau der Masterarbeit

Diese Masterarbeit gibt einen Überblick über vorhandene Graph-Cut-Verfahren und legt besonderen Fokus auf die kontinuierliche *Methode des erweiterten Residualgraphen (ERG)*. Diese wurde in einer Vorarbeit von Cremer in [19] entwickelt und konnte bereits zu Beginn gute experimentelle Ergebnisse erzielen. Das Verfahren weist jedoch einige Probleme und unbeantwortete Fragestellungen auf, die im Laufe dieser Arbeit thematisiert und korrigiert werden. Ein wesentlicher Nachteil liegt in der hohen Laufzeit des Verfahrens, die selbst bei kleinen Bildern bereits stark ansteigt. Zusätzlich berechnet das Verfahren keine optimale Lösung. Beides wird im Verlauf dieser Masterarbeit verbessert.

Kapitel 2 erläutert zunächst elementare Begrifflichkeiten aus der Graphentheorie. Anschließend werden das duale Minimalschnitt-Maximalfluss-Problem definiert und etablierte diskrete Lösungsmethoden vorgestellt. Darüber hinaus werden weitere Arbeiten präsentiert, welche das diskrete Graph-Cut-Problem nach verschiedenen Gesichtspunkten beleuchten und einige Lösungsansätze für vorhandene Probleme bieten.
Es folgt der Übergang in den kontinuierlichen Raum anhand von vorhandenen Verfahren in **Kapitel 3**. Dabei werden wesentliche Unterschiede zu den diskreten Algorithmen aufgeführt.
Im weiteren Verlauf der Arbeit wird die *Methode des erweiterten Residualgraphen* in **Kapitel 4** vorgestellt und deren Verhalten untersucht. Schließlich werden Verbesserungsvorschläge präsentiert und erläutert:

- Die Suche nach Pfaden im erweiterten Residualgraphen stellt einen wichtigen Bereich innerhalb des Verfahrens dar. Da in jeder Iteration ein Pfad gesucht wird, sollte ein Verfahren mit einer möglichst geringen Laufzeit gewählt werden. Dazu wird einerseits eine MATLAB-interne Methode betrachtet, die stets die kürzesten Pfade berechnet. Andererseits wird eine Erweiterung der Suche mithilfe von Suchbäumen geschildert.

- In der Vorarbeit wurde die Einhaltung einer Kapazitätsbedingung an den Fluss in der Quelle und Senke des Graphen vernachlässigt. Aufgrund des speziellen Aufbaus konnte eine abgewandelte Bedingung in diesen beiden Punkten entwickelt werden.

- Im Vergleich zu einer optimalen Lösung berechnet die *Methode des erweiterten Residualgraphen* häufig einen zu kleinen Maximalwert. Mithilfe von *Quasizyklen* wird ein Ansatz vorgestellt, der nicht nur ein approximatives, sondern ein optimales Ergebnis berechnen kann.

Abschließend wird das ERG-Verfahren mit anderen Algorithmen hinsichtlich Optimalität, Isotropie, das heißt der Richtungsunabhängigkeit, und Laufzeit in **Kapitel 5** verglichen und die Ergebnisse anhand synthetischer und realer Bilddaten vorgeführt.

Kapitel 2: Diskrete Graph-Cut-Verfahren

Dieses Kapitel beschreibt grundlegende Prinzipien der Graphentheorie. Dazu werden zunächst alle benötigten Strukturen definiert sowie eine nähere Erläuterung von Graph-Cut-Verfahren und dem Zusammenhang mit dem minimalen Schnitt und dem maximalen Fluss gegeben. Anschließend werden verschiedene Methoden zur Lösung von diskreten Graph-Cut-Problemen geschildert. Schließlich wird der Zusammenhang zur Bildsegmentierung hergestellt.

2.1 Grundlagen der Graphentheorie

Ein *Graph* beschreibt eine Struktur, welche verschiedene Objekte, im Folgenden als *Knoten* bezeichnet, miteinander verknüpft. Die Verbindungen werden im Zusammenhang der Graphentheorie *Kanten* genannt. Ein Graph ist nun wie folgt definiert:

Definition 1 (Graph [17])**.** Sei V eine Menge von *Knoten* und $E \subseteq \{\{v, w\} | v, w \in V\}$ eine Menge von *Kanten*. Dann heißt $G = (V, E)$ ein *(ungerichteter) Graph*. Außerdem gilt:

1. G heißt *gerichteter Graph*, wenn $E \subseteq V \times V$.
2. G heißt *gewichteter Graph*, wenn jeder Kante $e \in E$ eine Kapazität $c \in \mathbb{R}$ zugewiesen wird.
3. $G' = (V', E')$ heißt *Teilgraph* von G, wenn $V' \subset V$ und $E' \subset E$, wobei E' nur Knoten aus der Menge V' verbindet.

Im Laufe dieser Arbeit wird zudem davon ausgegangen, dass der Aufbau der Knoten- und Kantenmenge des Graphen unveränderlich sei. Im Zusammenhang mit Graph-Cut-Verfahren wird ein gewichteter und gerichteter Graph als sogenanntes *Netzwerk* aufgefasst. Innerhalb eines Netzwerkes werden zwei ausgezeichnete Knoten definiert: die *Quelle* und die *Senke*.

Definition 2 (Netzwerk [17])**.** Ein *Netzwerk* $N = (G, c, s, t)$ beinhaltet einen Graphen $G = (V, E)$ sowie eine Kostenfunktion $c : E \to \mathbb{R}$. Zusätzlich enthält N zwei spezielle Knoten $s \in V$ und $t \in V$, welche als *Quelle* und *Senke* bezeichnet werden.

In der Anwendung auf Bildsegmentierung sind unterschiedliche Modelle möglich, bei denen Quelle und Senke jeweils unterschiedliche Rollen spielen:

- Im *kantenbasierten Ansatz* werden die Quelle und Senke als Knoten innerhalb des Graphen definiert. Ein Beispiel ist links in Abbildung 2 dargestellt. Beide Knoten sind nur mit ihren Nachbarn durch Kanten verbunden.
- Der *regionenbasierte Ansatz* definiert die Quelle und die Senke zusätzlich zur Knotenmenge V. Sie werden wie im rechten Beispiel aus Abbildung 2 mit allen anderen Knoten durch Kanten verbunden.

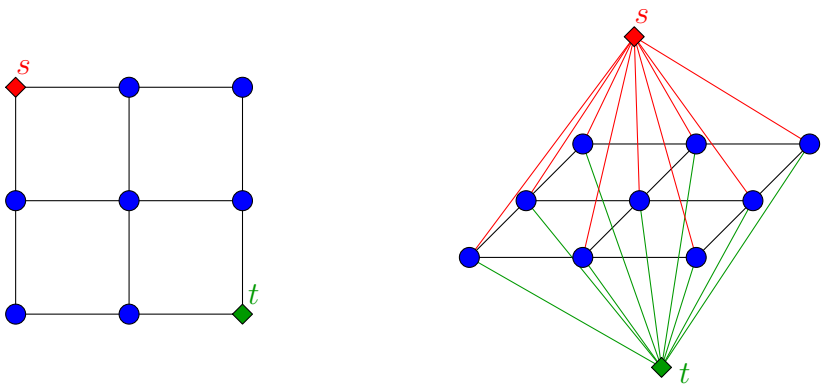


Abbildung 2. Graphstruktur sowie Rolle der Quelle und Senke für die Bildsegmentierung. Links sind die Quelle und die Senke Knoten des Graphen (*kantenbasierter Ansatz*). Rechts werden sie zusätzlich zum eigentlichen Graphen hinzugefügt und mit jedem Knoten über eine Kante verbunden (*regionenbasierter Ansatz*).

In dieser Masterarbeit werden die Quelle und Senke über den kantenbasierten Ansatz als Bildpunkte verstanden.
Innerhalb eines Netzwerkes N lässt sich ein *Fluss* f definieren. Dieser kann über alle Kanten des Graphen fließen. Besitzt eine Kante $e \in E$ keinen Fluss, so wird $f(e)$ auf null gesetzt. Flüsse fließen von der Quelle zur Senke und symbolisieren die Belastung der Kanten.

Definition 3 (Fluss [17])**.** Sei $N = (G, c, s, t)$ ein Netzwerk. Dann ist $f : V \times V \to \mathbb{R}$ ein *Fluss*, wenn die Bedingungen

$$f(u,v) \leq c(u,v)\ \forall u, v \in V, \qquad \textit{(Kapazitätsbedingung)} \tag{2}$$

$$f(u,v) = -f(v,u)\ \forall u, v \in V, \qquad \textit{(Antisymmetrie)} \tag{3}$$

$$\sum_{v \in V} f(u,v) = 0\ \forall u \in V \backslash \{s,t\} \qquad \textit{(Flusserhaltung)} \tag{4}$$

gelten. Der Fluss f wird als *zulässig* bezeichnet, wenn er die Kapazitätsbedingung (2) erfüllt. Zudem heißt der Fluss *gesättigt* in $(v, w) \in E$, wenn

$$f(v,w) = c(v,w). \tag{5}$$

Der *Wert des Flusses* ist definiert als

$$|f| = \sum_{v \in V} f(s,v). \tag{6}$$

Der Fluss ist auf jeder Kante des Graphen gegeben. Zudem erhält die zugehörige Rückkante über die Bedingung der Antisymmetrie (3) den entsprechenden negativen Wert. Die Menge des Flusses, der die Quelle verlässt, wird durch dessen Wert beschrieben.

Schließlich sorgt Kapazitätsbedingung dafür, dass der Fluss einer Kante ihre Kapazität nicht überschreiten kann. Ist eine Kante gesättigt, kann folglich kein weiterer Fluss über sie fließen. Zusätzlich erzwingt die Flusserhaltung, dass in einen Knoten ebenso viel Fluss einfließt wie ausfließt. Diese Bedingung gilt jedoch nicht für die Quelle und die Senke. Üblicherweise wird angenommen, dass die Quelle keinen Zufluss und die Senke keinen Ausfluss besitzt. Die Summe des aus der Quelle austretenden Flusses entspricht aufgrund der Flusserhaltung zudem der Summe des in die Quelle einfließenden Flusses.

Um sich innerhalb eines Graphen zu bewegen, werden an dieser Stelle *Wege* und *Pfade* durch G eingeführt. Sie bezeichnen Folgen von Knoten.

Definition 4 (Wege und Pfade [51])**.** Sei $N = (G, c, s, t)$ ein Netzwerk mit $G = (V, E)$. Ein *Weg* ist eine geordnete Knotenfolge $(v_1, \ldots, v_k)$, so dass $(v_i, v_{i+1}) \in E$ für alle $i < k$. Ein *Pfad* ist ein Weg $(v_1, \ldots, v_k)$ mit $v_i \neq v_j$ für alle $i < j \leq k$.

Ein Weg kann somit Knoten mehrfach verwenden. Bei einem Pfad darf kein Knoten doppelter Bestandteil der Folge sein. Existiert ein Pfad in einem Graphen, dessen Kanten nicht gesättigt sind und entlang derer der Fluss weiter erhöht werden kann, ohne die jeweilige Kapazität zu überschreiten, so spricht man von einem *augmentierenden Pfad*:

Definition 5 (Augmentierender Pfad [51])**.** Sei $N = (G, c, s, t)$ ein Netzwerk mit $G = (V, E)$. Eine Folge von Knoten $P = (v_1, \ldots, v_n), v_i \in V, i \in \{1, \ldots, n\}$ heißt *augmentierender Pfad*, falls gilt:

1. $(v_{i-1}, v_i) \in E$ und $f(v_{i-1}, v_i) < c(v_{i-1}, v_i)$ für alle Vorwärtskanten $(v_{i-1}, v_i) \in P$
2. $(v_i, v_{i-1}) \in E$ und $f(v_i, v_{i-1}) > 0$ für alle Rückwärtskanten $(v_i, v_{i-1}) \in P$.

Wichtig für die in Abschnitt 2.3 erläuterten Algorithmen ist die Betrachtung eines sogenannten *Residualgraphen*. Dieser beschreibt, wie viel Fluss bei gegebenen Kapazitäten durch den Graphen fließen kann. Dabei gibt eine *Residualkapazität* an, wie viel zusätzlicher Fluss durch den Graphen fließen kann, ohne die Kapazitätsbedingung zu verletzen.

Definition 6 (Residualgraph [17])**.** Sei $G = (V, E)$ ein Graph mit der Kostenfunktion $c : V \times V \to \mathbb{R}$ und einem Fluss $f : V \times V \to \mathbb{R}$. Dann ist $G_f = (V, E_f)$ der *Residualgraph* von G zum Fluss f.
Die Kantenmenge des Residualgraphen ist mit $E_f := \{(v, w) \in V \times V : c_f(v, w) > 0\}$ gegeben. Dabei wird $c_f(v, w) := c(v, w) - f(v, w)$ als *Residualkapazität der Kante* (v, w) bezeichnet.

2.2 Minimaler Schnitt und Maximaler Fluss

Die Grundlage für Graph-Cut-Verfahren sind sogenannte *s-t-Schnitte* in Netzwerken. Sie unterteilen den Graphen G so in zwei Bereiche, dass Quelle und Senke voneinander getrennt werden. Die Knoten des Graphen werden dabei wie in Abbildung 3 gruppiert, wodurch eine Menge durchtrennter Kanten entsteht.

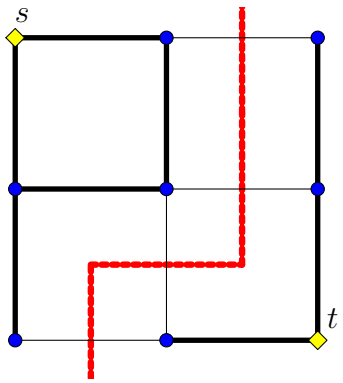


Abbildung 3. Darstellung eines Graphen mit Quelle s und Senke t (gelb). Die Höhe der Kantenkosten ist durch die Dicke der Linien symbolisiert. Der Schnitt durch den Graphen (rot) führt somit durch die Kanten mit den geringsten Kapazitäten.

Definition 7 (s-t-Schnitt)**.** Sei $N = (G, c, s, t)$ ein Netzwerk mit $G = (V, E)$, $c : E \to \mathbb{R}$ und $s, t \in V$. Dann besteht ein *s-t-Schnitt* aus einer Teilmenge $S \subseteq V$ mit $s \in S$ und $t \notin S$ sowie einer Menge durchtrennter Kanten $\hat{E} = \{(v, w) \in E | v \in S, w \in V \backslash S\}$. Die *Kosten* eines s-t-Schnittes sind definiert als

$$c(S) := \sum_{\substack{e=(v,w),\\ v\in S, w\in V\backslash S}} c(e) \tag{7}$$

Ein s-t-Schnitt mit minimalen Kosten gegenüber allen möglichen s-t-Schnitten wird als *minimaler Schnitt* bezeichnet. In engem Zusammenhang zu einem s-t-Schnitt steht der Fluss, da dieser durch die Kapazitäten des Graphen beschränkt wird. Ist der Wert eines Flusses maximal (*Maximalfluss*), so lässt sich kein Pfad im Residualgraphen von der Quelle zur Senke finden, wodurch f einen Schnitt induziert. Tatsächlich lässt sich zeigen, dass der maximal mögliche Fluss in G dem Wert des minimalen s-t-Schnittes entspricht:

Satz 1 (Maximalfluss-Minimalschnitt-Theorem [17, Kapitel 26, Theorem 26.7])**.** Sei $N = (G, c, s, t)$ ein Netzwerk mit einem Fluss f^*. Zudem bezeichne $c(S^*)$ den Wert eines minimalen s-t-Schnittes S^*. Dann sind folgende Aussagen äquivalent:

1. f^* ist ein maximaler Fluss.
2. Der Residualgraph G_{f^*} enthält keine Pfade.
3. $|f^*| = c(S^*)$

Beweisskizze nach [17].
(1) $\Rightarrow$ (2):
Kann im Residualgraphen ein Pfad gefunden werden, so entspricht dieser einem augmentierenden Pfad in G, entlang dessen der Fluss weiter vergrößert werden kann. Dies widerspricht der Aussage, dass der Fluss bereits maximal ist.

(2) ⇒ (3):
Sei S die Menge der Knoten, die von der Quelle aus erreichbar sind und T beinhalte alle verbliebenen Knoten. Wenn keine Pfade in G_f enthalten sind, dann ist $c(v, w) = f^*(v, w)$ für alle $v \in S, w \in T$, da andernfalls $(v, w) \in E_{f^*}$.

(3) ⇒ (1):
In [17, Kapitel 26, Lemma 26.5 und Korollar 26.6] wurde gezeigt, dass $|f^*| \leq c(S^*)$ für alle s-t-Schnitte S^*. Die Bedingung $|f^*| = c(S^*)$ impliziert nun, dass f^* ein maximaler Fluss ist, da f^* die Kapazität nicht überschreiten kann. □

Um den minimalen s-t-Schnitt zu berechnen, kann ein maximaler Fluss gefunden werden, welcher ein äquivalentes Ergebnis liefert. Das Maximalflussproblem ist zudem in der Regel leichter zu lösen. In Abschnitt 2.3 werden einige Methoden zur Lösung dieses Problems beschrieben. Als Spezialfall des Dualitätstheorems der linearen Optimierung [39, 43] wird das Maximalflussproblem im Weiteren als das *duale Problem* des *primalen Minimalschnittproblems* bezeichnet.

2.3 Diskrete Graph-Cut-Verfahren

In diesem Abschnitt werden drei der bekanntesten Verfahren zur Lösung des Maximalflussproblems erläutert:

- Die *Augmenting-Path-Methode* von Ford und Fulkerson [23],
- die *Preflow-Push-Methode* von Goldberg und Tarjan [26],
- sowie eine Erweiterung der Augmenting-Path-Methode mit zwei Suchbäumen von Boykov und Kolmogorov [11].

Alle Verfahren liefern das gleiche Endergebnis und denselben Fluss, beruhen allerdings auf unterschiedlichen Ansätzen und unterscheiden sich in der Laufzeit. Anschließend werden weitere Arbeiten in Zusammenhang mit Graph-Cut-Methoden vorgestellt.

2.3.1 Ford-Fulkerson-Methode

Die Methode von Ford und Fulkerson [23] beschreibt eine Möglichkeit zur Lösung des Maximalfluss-Problems. Dabei wird ein Pfad durch einen gegebenen Graphen G von der Quelle s zur Senke t gesucht, entlang dessen der Fluss weiter vergrößert werden kann, ohne die Kapazitätsbeschränkungen des Flusses (2) zu verletzen. Daher ist das Verfahren als *Augmenting-Path-Methode* bekannt.

Ausgehend von einem zulässigen initialen Fluss (beispielsweise $f(v, w) = 0 \, \forall (v, w) \in E$) werden mithilfe des Residualgraphen G_f iterativ augmentierende Pfade P von der Quelle zur Senke gesucht, welche einen zusätzlichen Fluss durch den Graphen erlauben. Entlang dieser Pfade wird der Fluss um den maximal möglichen Wert $\delta_P = \min_{e \in E(P)} c_f(e)$ erhöht, der die Kapazitätsbedingung (2) nicht verletzt. Das Verfahren terminiert, wenn

kein s-t-Pfad im Residualgraphen zu finden ist. Bei ganzzahligen Kantenkapazitäten ist das Auffinden eines maximalen Flusses in endlich vielen Schritten garantiert [23]. Eine detaillierte Beschreibung der Methode zeigt Algorithmus 1.

Die Methode von Ford und Fulkerson weist eine pseudopolynomielle Laufzeit auf. Werden die augmentierenden Pfade stets als die kürzesten möglichen Pfade gewählt, so lässt sich die Laufzeit auf $\mathcal{O}(VE^2)$ verbessern [22]. Dieser Ansatz für die Verringerung der Laufzeit wird auch als *Edmonds-Karp-Algorithmus* bezeichnet.

Algorithmus 1 Ford-Fulkerson-Methode [17]

Eingabe: Graph $G = (V, E)$, Quelle und Senke s, t
1: Initialisiere den Fluss mit $f = 0$
2: **Solange** ein Pfad P von s nach t in G_f existiert
3: $\quad \delta_P = \min(c_f(u, v) : (u, v) \in P)$ ▷ *Augmentiere entlang P*
4: $\quad$ **Für** jede Kante $(u, v) \in P$
5: $\quad\quad f(u, v) = f(u, v) + \delta_P, f(v, u) = -f(u, v)$
Rückgabe: f

2.3.2 Preflow-Push-Methode

Einen andereren Ansatz zur Lösung des Maximalfluss-Problems bietet die Methode von Goldberg und Tarjan [26], bekannt als *Preflow-Push-* oder *Push-Relabel-Methode*. Zum Verständnis der Methode werden die Definitionen des *Überschusses* und eines *Präflusses (engl.: preflow)* benötigt. Der Überschuss beschreibt dabei die Differenz der in einen Knoten eingehenden und auslaufenden Flüsse. In der Methode von Ford und Fulkerson ist dieser aufgrund der Flusserhaltung in allen Knoten bis auf Quelle und Senke stets null (siehe Definition 3).

Definition 8 (Überschuss [26])**.** Sei $G = (V, E)$ ein Graph und $f : E \to \mathbb{R}$ ein Fluss. Dann heißt $\epsilon : V \to \mathbb{R}$ mit

$$\epsilon(u) = \sum_{v \in V} f(v, u) \tag{8}$$

Überschuss von f.

Im Gegensatz zur Methode von Ford und Fulkerson beruht dieser Ansatz auf einer lokalen Betrachtung des Residualgraphen, bei der Kanten einzeln untersucht werden. Jeder Knoten wird dabei als eine Art Behältnis interpretiert, welches unendlich viel Fluss aufnehmen und in einem nächsten Schritt an benachbarte Knoten weiterleiten kann. Auf Basis dieser Interpretation verwendet die Methode die Verallgemeinerung des in Abschnitt 2.1 definierten Flusses.

Definition 9 (Präfluss [26])**.** Sei $f : E \to \mathbb{R}$. Erfüllt f die folgenden Bedingungen, so wird f als *Präfluss* bezeichnet:

1. $f(e) \leq c(e)$ $\quad \forall e \in E$ (Kapazitätsbedingung),
2. $f(e) = -f(e)$ $\quad \forall e \in E$ (Antisymmetriebedingung),
3. $\epsilon(v) \geq 0$ $\quad \forall v \in V \backslash \{s, t\}$ (Überschuss).

Anders als der in Definition 3 eingeführte Fluss, fordert ein Präfluss keine Flusserhaltung, sondern lediglich einen nichtnegativen Überschuss. Die Methode terminiert, wenn die Flusserhaltung erfüllt ist.

Zusätzlich wird jedem Knoten eine „Höhe“ zugeordnet. Der Präfluss kann nur dann zu einem anderen Knoten weitergeleitet werden, wenn dieser eine geringere Höhe besitzt, der Fluss somit bildlich gesehen abwärts fließt.

Definition 10 (Höhenfunktion)**.** Sei $N = (G, c, s, t)$ ein Netzwerk mit $G = (V, E)$. Dann heißt $h : V \to \mathbb{N}$ *Höhenfunktion*, wenn $h(s) = |V|, h(t) = 0$ und $h(u) \leq h(v) + 1$ für alle Residualkanten (u, v).

Alle weiteren Knoten bis auf die Quelle werden mit einer Höhe von null initialisiert und iterativ erhöht. Der Algorithmus besteht aus zwei möglichen Operationen: der *Push*- und der *Relabel-Operation*.

Die *Push*-Operation beschreibt die Weiterleitung des Flusses von einem Knoten u zu einem Nachbarn v. Sie wird ausgeführt, wenn dem Folgeknoten exakt eine Höheneinheit weniger zugewiesen und die Residualkapazität positiv ist, was einem Überfluss entspricht. Algorithmus 2 beschreibt die Aktualisierung des Präflusses und des Überschusses während der *Push*-Operation.

Algorithmus 2 *Push*(u, v)-Operation [17]

Eingabe: Kante $(u, v) \in E$

Falls $c_f(u, v) > 0$ und $h(u) = h(v) + 1$
  $\delta_f(u, v) = \min(e(u), c_f(u, v))$
  $f(u, v) = f(u, v) + \delta_f(u, v), f(v, u) = -f(u, v)$
  $\epsilon(u) = \epsilon(u) - \delta_f(u, v), \epsilon(v) = \epsilon(v) + \delta_f(u, v)$

Ist keine entsprechende Kante (u, v) vorhanden, welche die Bedingungen der *Push*-Operation erfüllt, werden die gefordeten Bedingungen der *Relabel*-Operation geprüft. Existiert eine Kante (u, v) mit positiver Residualkapazität sowie einem Höhenanstieg in Richtung des Knotens v, so wird das Niveau des Knotens u erhöht, um im nächsten Schritt erneut eine *Push*-Operation zu ermöglichen. Algorithmus 3 zeigt das Vorgehen sowie die nötigen Bedingungen.

Algorithmus 3 *Relabel*(u)-Operation [17]

Eingabe: Knoten $u \in V$

Falls $(u, v) \in E_f, c_f(u, v) > 0$ und $h(u) \leq h(v)$
    $h(u) = 1 + \min(h(v) : (u, v) \in E_f)$

Die vollständige Preflow-Push-Methode ist in Algorithmus 4 beschrieben. Zunächst werden die Höhenfunktion und der Überschuss initialisiert und der Präfluss in allen Knoten, die eine Verbindung zur Quelle aufweisen, berechnet. $A = (a_{ij})_{i,j=1}^{n}$ bezeichnet dabei die Adjazenzmatrix von G mit $a_{i,j} = \begin{cases} 1, & \text{wenn } (i,j) \in E \\ 0 & \text{sonst} \end{cases}$.
Anschließend wird die Durchführbarkeit der beiden Operationen für alle Knoten geprüft und die Höhe sowie der Präfluss iterativ erhöht und aktualisiert [17].

Im Vergleich zu den in Abschnitt 2.3.1 vorgestellten Ford-Fulkerson- und Edmonds-Karp-Algorithmen besitzt die Preflow-Push-Methode eine schnellere asymptotische Laufzeitklasse von $\mathcal{O}(VE \log(V^2/E))$ [26].

Algorithmus 4 Push-Relabel-Methode [17]

Eingabe: Netzwerk $N = (G, c, s, t)$
Initialisiere Preflow $f := 0$, Überschuss $\epsilon := 0$ und Höhe $h := 0$ sowie $h(s) = |V|$
Für jeden Knoten u mit $A(s, u) = 1$
    $f(s, u) = c(s, u), f(u, s) = -c(s, u)$
    $\epsilon(u) = c(s, u), \epsilon(s) = \epsilon(s) - c(s, u)$
Solange eine *Push*- oder *Relabel*-Operation durchführbar ist
    Führe *Push*- oder *Relabel*-Operation durch
Rückgabe: f

2.3.3 Methode von Boykov und Kolmogorov

Auf Basis der Methode von Ford und Fulkerson entwickelten Boykov und Kolmogorov im Jahr 2004 einen Algorithmus, der in der Praxis bis zu fünfmal schneller arbeitet [11], da die Pfadsuche durch die Verwendung zweier Suchbäume S und T optimiert wird. Diese werden iterativ aktualisiert und müssen nicht in jedem Schritt neu berechnet werden. Die zuvor präsentierte Augmenting-Path-Methode benötigt im Gegensatz dazu in jeder Iteration eine erneute Pfadsuche. Ein *Baum* bezeichnet dabei einen Graphen, in dem zwischen je zwei beliebigen Knoten $v, w \in V$ ein Weg existiert und der keine geschlossenen Pfade enthält. Ausgehend von einem *Wurzelknoten* sind alle anderen Knoten zu erreichen [7].

Die Knotenmenge wird in *aktive*, *passive* und *freie* Knoten aufgeteilt. Letztere sind keinem Suchbaum zugeordnet. Aktive Knoten formen eine äußere Grenze der Bäume.

Ausgehend von ihnen können die Suchbäume weiter wachsen. Hingegen werden innere Knoten als passiv bezeichnet, da sie keinen Beitrag zum Wachstum leisten können. Für jeden Knoten wird zusätzlich markiert, von welchen Nachbarn aus er erreicht wurde (gespeichert in der Variable *parent*). Darüber hinaus speichert die Variable *tree*, in welchem Suchbaum sich der Knoten befindet oder ob es sich um einen freien Knoten handelt.

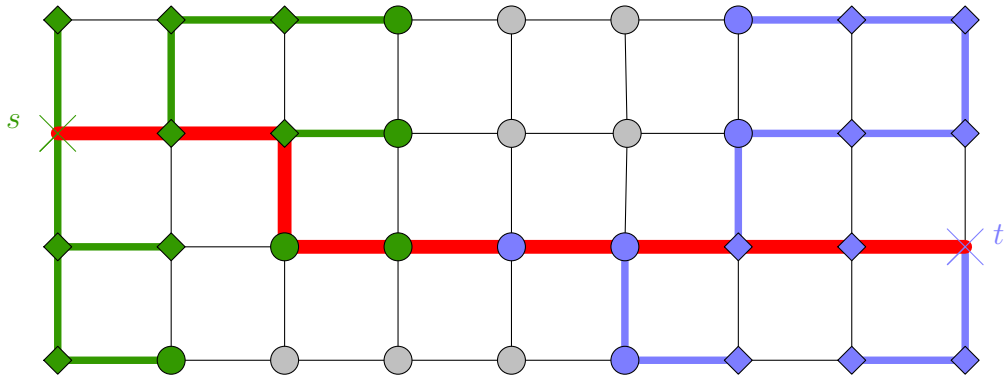


Abbildung 4. Beispiel der Suchbäume S (grün) und T (blau) am Ende der Wachstumsphase. Aktive Knoten sind als Kreis, passive Knoten als Raute dargestellt. Freie Knoten sind durch graue Kreise symbolisiert. Die Quelle s und die Senke t wurden mit einem Kreuz markiert. Die Wachstumsphase ist in diesem Beispiel beendet, da sich zwei aktive Knoten der beiden Suchbäume treffen und somit ein Pfad (rot) von der Quelle zur Senke gefunden werden kann. Die Abbildung wurde erstellt nach [11, Seite 9].

Während der sogenannten *Wachstumsphase* wird nach aktiven Knoten $p \in V$ gesucht, denen noch kein Suchbaum zugeordnet ist. Ist p durch eine Kante mit positiver Residualkapazität zu einem Nachbarknoten $q \in V$ verbunden, der einem der Bäume S oder T zugeordnet ist, so wird p demselben Suchbaum wie q zugewiesen. Es wird solange über die aktiven Knoten iteriert, bis keine weiteren vorhanden sind oder bis zwei aktive Knoten verschiedener Bäume sich treffen. Ist dies der Fall, wurde ein augmentierender Pfad P gefunden. Analog zur Methode von Ford und Fulkerson wird der Fluss entlang dieses Pfades um den maximal möglichen Wert δ erhöht, der die Kapazitätsbedingungen nicht verletzt (*Augmentierungsphase*).

Innerhalb eines Suchbaums werden nur diejenigen Knoten verbunden, deren Residualkapazität positiv ist. Durch die Sättigung einer oder mehrerer Kanten kann es somit passieren, dass die Suchbäume in Teilbäume zerfallen. Die neu entstandenen Wurzelknoten werden als *Waisen* bezeichnet. Für diese werden im Anschluss neue Elternknoten gesucht, um die entstandenen Teilbäume erneut mit S oder T zu verknüpfen (*Adoptionsphase*) [11].

Die grobe Struktur der Methode zeigt Algorithmus 5. Eine vollständige und detaillierte Beschreibung kann in Algorithmus 12 in Anhang A.1 gefunden werden.
Das Zusammenfügen der Suchbäume hat den Nachteil, dass nicht immer die kürzesten Pfade gefunden werden. Im ungünstigsten Fall kann der Algorithmus von Boykov und Kolmogorov demnach eine höhere Laufzeit als die Methode von Ford und Fulkerson aufweisen. Zusätzlich sind durch den Ansatz einige Stellen gegeben, an denen die Programmierer freie Wahl in der Vorgehensweise haben, wie beispielsweise in der Reihenfolge des Abrufs der aktiven Knoten. Um zumindest im ersten Schritt den kürzesten Pfad durch den Graphen zu finden, wird für den Abruf der aktiven Knoten ein *First-In–First-Out-Verfahren* gewählt [11]. Bei Anwendung des Verfahrens auf dichtere Graphen oder 3D-Probleme verliert es jedoch an Effizienz. Im schlechtesten Fall weist es zudem eine Laufzeitkomplexität von $\mathcal{O}(VE^2|S|)$ auf [11].

Algorithmus 5 Algorithmus von Boykov und Kolmogorov (kurz) [11]

1: Initalisiere $S = \{s\}, T = \{t\}, A = \{s, t\}$: Aktive Knoten, $O = \emptyset$: Waisen
2: **Solange** „wahr“
3: Lasse S oder T *wachsen*, um einen augmentierenden Pfad P von s nach t zu finden
4: **Falls** $P = \emptyset$
5: Terminiere
6: *Augmentiere* auf P
7: *Adoptiere* Waisen

2.3.4 Capacity Scaling für Graph-Cut-Verfahren

Aufgrund der fehlenden Effizienz des im vorigen Abschnitt vorgestellten Verfahrens von Boykov und Kolmogorov bei komplexeren Problemen [31] entwickelten Juan und Boykov [31] im Jahr 2007 eine Verbesserung auf Basis eines hierarchischen Multiskalenansatzes. Das Verfahren beginnt zunächst mit einer groben Auflösung, in der ein Maximalflussproblem nur Kanten der Menge $E_{\Delta_k} \subseteq E$ mit Kapazitäten oberhalb eines Schwellwerts Δ_k berücksichtigt. Nach Beendigung einer Skalierung Δ_k geht der Algorithmus zu einer feineren Skalierung über, indem weitere Kanten zu der Menge E_{Δ_k} hinzugefügt werden. Diese liegen oberhalb des neuen Schwellwerts $\Delta_{k-1} < \Delta_k$ der Skalierung. Die neue Lösung des Maximalflussproblems verwendet nun die unbenutzten Pfade der geringeren Kapazität Δ_{k-1}. Die feinste Skalierung liefert schließlich das globale Maximum des ursprünglichen Optimierungsproblems.

Der Multiskalenansatz ermöglicht ein schnelles Anwachsen des maximalen Flusses, auch wenn nicht in jedem Schritt ein kürzester Pfad berechnet wird. Wird die Kapazitätsskalierung auf das Verfahren von Boykov und Kolmogorov [11] angewandt, so lässt sich dessen Komplexität von $\mathcal{O}(VE^2|S|)$ auf eine schwach polynomielle Komplexität von $\mathcal{O}(V^2E^2 \log(U))$ reduzieren, wobei U das größte Kantengewicht bezeichnet. Bei diesem Algorithmus wird von ganzzahligen Kapazitäten ausgegangen.

Um die Suchbäume aus dem Verfahren von Boykov und Kolmogorov [11] in jeder Skalierung zu erhalten, werden einige Veränderungen am Ansatz vorgenommen. Zum Ende einer jeden Skalierung terminiert das Verfahren von Boykov und Kolmogorov [11], so dass die Menge der aktiven Knoten leer ist. Um die Suchbäume und die aktiven Knoten für die nächste Skalierung weiter verwenden zu können, werden sie reaktiviert. Dies kann beispielsweise durch Listen geschehen, welche die Knoten an den äußeren Grenzen der Suchbäume speichern.

Der Ansatz ist vergleichsweise effizient, da die Wiederverwendung der Suchbäume garantiert wird, jedoch ist die zusätzliche Komplexität nicht zu vernachlässigen. Dies zeigt sich vor allem im Vergleich des Verfahrens von Boykov und Kolmogorov [11] mit der zusätzlichen Verwendung einer Kapazitätsskalierung bei recht einfachen Beispielen wie einem zweidimensionalen Graphen mit einer 4er-Nachbarschaft. Hier ist der Algorithmus von Boykov und Kolmogorov [11] weiterhin schneller [31]. Der Übergang zu komplexeren Beispielen mit dichteren Gittern und Rauschen zeigt jedoch den Zeitvorteil des kombinierten Ansatzes. Bei dreidimensionalen Gittern kann er beispielsweise eine Beschleunigung bis zu einem Faktor von sechs aufweisen [31]. Um die Laufzeit weiter zu beschleunigen, können schließlich approximative Lösungen der gröberen Skalierungen verwendet werden.

2.3.5 Voronoi-basiertes Preflow-Push-Verfahren

Eine weitere Möglichkeit zur Verbesserung der Laufzeit von Graph-Cut-Verfahren bietet der Ansatz von Arora et al. [4]. Wie in Abschnitt 2.3.2 wird auch hier für alle Knoten ein Überschuss definiert. Anders als bei den vorherigen Methoden wird der Fluss auf den Kanten, die mit der Quelle beziehungsweise Senke verbunden sind, mit ihren Kapazitäten initialisiert. Der Fluss auf allen anderen Kanten wird zu Beginn auf null gesetzt. Die Höhenfunktion wird für alle Knoten außer in Quelle und Senke mit ihrem Überschuss definiert. In dieser neuen Variante seien nun alle Knoten mit positiven Überschussen als Quellen, alle mit negativen Überschussen als Senken bezeichnet [4].

Für den Algorithmus werden nun in jedem Schritt sogenannte *Voronoi-Regionengraphen* erstellt. Dafür sind die Quellen- und Senkenknoten in verschiedenen Clustern angeordnet. Die Höhenlabel bezeichnen hier den kürzesten Abstand eines Senkenknotens zu dessen Clustergrenze. Die Zuweisung der Höhenlabel terminiert, wenn allen Quellenknoten ein Label zugewiesen ist. Die entstehenden Untergraphen werden als Voronoi-Regionengraphen bezeichnet. Für jeden dieser Untergraphen wird ein Fluss hinzugefügt. Dabei werden zuerst die Knoten mit den größten Höhenwerten prozessiert. Erreicht der Fluss die Grenzen eines Senkenclusters, fließt der Fluss in einer zweiten *Push*-Phase in das Cluster hinein. Ist aller Überschuss der Knoten aufgehoben, so schrumpft das jeweilige Cluster. Sind jedoch vorher bereits alle Knoten prozessiert, verschwindet es. Der grobe Ablauf des Verfahrens ist in Algorithmus 6 zu sehen. Für eine detailliertere Fassung siehe [4].

Algorithmus 6 Voronoi-basiertes Preflow-Push-Verfahren [4]

1: Erstelle Voronoi-Regionengraphen mit kürzesten Distanzen der Senkencluster
2: **Für** alle Voronoi Regionengraphen mit Quellen
3: Erhöhe den Fluss von den Quellen zu den Senkenclustern innerhalb jeder Voronoi-Region
4: Erhöhe den Fluss innerhalb der Senkencluster
5: Erstelle neue Voronoi Regionengraphen um die verbliebenen Senkencluster

In der Praxis weist das Verfahren eine streng polynomielle Laufzeitkomplexität auf und arbeitet zwei- bis dreimal schneller als das Verfahren von Boykov und Kolmogorov [10].

2.3.6 Effiziente planare Graph-Cut-Verfahren

Lange Zeit wurde das Verfahren von Boykov und Kolmogorov [11] als die schnellste Methode für Anwendungen im Bereich der Computer Vision angesehen. Im Jahr 2009 präsentierten Schmidt et al. [46] einen Ansatz für planare Graph-Cut-Verfahren mit einer annähernd linearen Laufzeit, die für ein „worst-case“-Szenario schneller arbeitet als das Verfahren von Boykov und Kolmogorov [46].

Planare Netzwerke können in den $\mathbb{R}^2$ eingebettet werden, wodurch die Kanten des Graphen die Ebene in verschiedene *Zellen* F aufteilen. Innerhalb eines planaren Graphen $G = (V, E)$ schneiden sich keine Kanten. Zusätzlich besitzt jede Kante $e \in E$ eine linke Zelle $f_l(e) \in F$ und eine rechte Zelle $f_r(e) \in F$. Mithilfe dieser Definitionen lässt sich ein duales Netzwerk $G^* = (F, E^*, c, s^*, t^*)$ festlegen, dessen duale Kanten $e^* = (f_l(e), f_r(e))$ die jeweilige linke mit der rechten Zelle der Kante verbinden. Die duale Quelle und Senke sind zufällige Zellen in der Nähe der primalen Knoten $s, t \in V$. Ähnlich wie die zuvor beschriebenen Verfahren sucht der Algorithmus zulässige Pfade von der Quelle zur Senke und sättigt jeweils den äußersten linken Pfad. Ein Spannbaum T wird wie beim Verfahren von Boykov und Kolmogorov [11] verwendet, um die Pfadsuche zu beschleunigen. Zusätzlich zur iterativen Aktualisierung von T wird ein dualer Spannbaum T^* der Zellen F eingeführt, der alle Kanten von $E \backslash T$ beinhaltet. Das vollständige Verfahren ist in Algorithmus 7 gegeben.

Es konnte gezeigt werden, dass die in Algorithmus 7 beschriebene Methode ein Maximalflussproblem mit einer Laufzeit von $\mathcal{O}(N \log N)$ lösen kann. Diese Laufzeitbeschleunigung beruht vor allem auf der Reduktion der Größe der benötigten Datenstrukturen [46].

Algorithmus 7 Algorithmus zur Berechnung effizienter planarer Graph-Cuts [46]

Eingabe: $G = (V, E), F, f_l \in F, f_r \in F, c : E \to \mathbb{R}, s \in V, t \in V$
Ausgabe: Maximalfluss $f : E \to \mathbb{R}$

```
Initialisiere f = 0
Solange ein zulässiger Pfad von s nach t existiert
    Sättige den äußersten linken Pfad von s nach t:
    Wiederhole
        Augmentiere Pfad und aktualisiere f
        Sei d = (u, v) die nächste Kante zu t, die gesättigt wird
        Sei d* = (f1, f2) die zu d duale Kante
        Sei e* = (f2, f3) die Elternkante von d* in T*
        Sei e = (x, y) dessen zugehörige primale Kante
        T* = T* + {(f2, f1)} − {(f2, f3)}, T = T − {d} + {e}
        Falls f1 Nachkomme von f2 in T*
            Gib f zurück
        Invertiere Kanten in T über Pfad von x nach u
    bis T, T* keine Bäume mehr sind
Rückgabe: f
```

2.4 Berechnung von Geoschnitten auf Basis von Graph-Cut-Verfahren

Zur Berechnung von minimalen Oberflächen stellten Boykov und Kolmogorov [10] 2003 eine Kombination aus Geodesic Active Contours und Graph-Cut-Verfahren vor, um die Vorteile beider Methoden zu vereinen. Anstelle einer Pfad-Metrik

$$|p_{AB}| = \sum_{e \in p_{AB}} c_e, \quad A, B \in V \tag{9}$$

des Pfades $p_{AB} \in E$ zur Berechnung der Kosten eines Schnittes wird hier eine dazu duale Schnitt-Metrik eingeführt. Diese berechnet die Kosten des berechneten Schnittes $S \subset V$:

$$|S|_G = \sum_{e \in S} c_e. \tag{10}$$

Aufgrund der geometrischen Interpretation der Schnitt-Metrik als die Länge des Schnittes S besitzt $|S|_G$ die Eigenschaften einer Metrik, basierend auf dem Nachbarschaftssystem und den Kapazitäten des Graphen G. Im Gegensatz zu der zuvor vorhandenen Pfad-Metrik ist die Schnitt-Metrik nicht auf eindimensionale Pfade (Konturen) beschränkt und kann die euklidische und riemannsche Metrik approximieren. Sie bietet zudem den Vorteil einer Invarianz gegenüber der Vertauschung von Quelle und Senke.

2.5 Graph-Cut-Verfahren basierend auf linearen Programmen

Graph-Cut-Verfahren liefern eine gute Grundlage, um sogenannte *Metric-Labeling-Probleme (ML-Probleme)* zu lösen. Dabei soll jedem Objekt einer Menge V ein Label aus einer gegebenen Menge $\mathcal{L}$ zugewiesen werden. Mithilfe eines Graphen $G = (V, E)$ mit Kantengewichten $c_{p,q}$ für $(p, q) \in E$ kann die Lösung eines Maximalflussproblems jedem Knoten ein entsprechendes Label zuordnen [36]. Viele bekannte Lösungsansätze beruhen dabei auf der kombinatorischen Optimierung (beispielsweise in [14]). Ein alternativer Ansatz basiert auf linearen Programmen, jedoch haben diese häufig den Nachteil einer schlechten Laufzeit. Eine Möglichkeit zur Verbesserung lieferten Komodakis und Tziritas [36] mithilfe von Prinzipien aus der Dualitätstheorie von linearen Programmen. Zu diesem Zweck wird das primale ML-Problem in seine Dualform überführt, woraufhin primal-duale Algorithmen hergeleitet werden.

Die vorgestellten Algorithmen liefern insbesondere eine Generalisierung von Graph-Cut-Verfahren zur Lösung von ML-Problemen und somit eine mögliche Anwendung auf allgemeinere Probleme wie Markov-Netzwerke mit nichtmetrischen Potentialen. In verschiedenen Anwendungsbeispielen wie der Bildrestauration oder der Schätzung eines optischen Flusses liefern die entwickelten Algorithmen ebenso gute Ergebnisse wie etablierte Verfahren mit Lösungsmethoden der kombinatorsichen Optimierung [36].

2.5.1 Netzwerkprobleme nach Bertsekas [6]

Maximalfluss- beziehungsweise Minimalschnittproblemen können als lineare Programme aufgefasst werden [6]. Das Problem des minimalen Schnittes wird als Minimierung der Kantenkosten a_{ij} mithilfe des Flusses x_{ij} ausgelegt:

$$\min_{a} \sum_{(i,j)\in E} a_{ij}x_{ij}, \quad \text{so dass} \tag{11}$$

$$\sum_{\{j|(i,j)\in E\}} x_{ij} - \sum_{\{j|(j,i)\in E\}} x_{ji} = s_i \quad \forall i \in V \tag{12}$$

$$b_{ij} \leq x_{ij} \leq c_{ij} \quad \forall (i,j) \in E. \tag{13}$$

Der Fluss einer Kante $(i, j) \in E$ wird von Bertsekas als x_{ij} bezeichnet. Die Skalare b_{ij} und c_{ij} stellen Flussbegrenzungen der Kante $(i, j) \in E$ dar und s_i beschreibt den Vorrat eines Knotens $i \in V$. Letzterer entspricht dem in Abschnitt 2.3.2 vorgestellten Überschuss. Die Flusserhaltung wird in diesem Problem über Gleichung (12) beschrieben. Für die bereits geschilderten Verfahren nimmt s_i in jedem Knoten außer in Quelle und Senke den Wert null an. Des Weiteren wird die Kapazitätsbedingung in Gleichung (13) erweitert, das heißt der Fluss ist nach oben und unten beschränkt.

Bertsekas stellt verschiedene Ansätze vor, um dieses und ähnliche Netzwerkprobleme zu lösen. Eine Kategorie bilden die *primalen Kostenverbesserungen*, bei denen die Kosten durch die Konstruktion zulässiger Flussvektoren $x = (x_{ij})_{(i,j)\in E}$ iterativ verbessert

werden. Das dazu duale Problem der *dualen Kostenverbesserung* beinhaltet die Maximierung von *Preisvariablen* $p = (p_i)_{i \in V}$. Diese werden durch einen sogenannte *komplementären Schlupf* mit den Kosten a_{ij} verbunden. Das Problem wird so interpretiert, dass eine Menge von Personen verschiedene Objekte zu einem gewissen Preis erwerben möchte und dabei eine wirtschaftliche Vorgehensweise verfolgen. Jede Person i versucht ein Objekt j mit dem optimalen Wert zu erhalten. Demnach wird das Maximum der Differenz des Werts a_{ij} und des Preises p_j betrachtet [6]:

$$q_i(p) = \max_{j \in \{j|(i,j) \in E\}} \{a_{ij} - p_j\}. \tag{14}$$

Gilt diese Gleichung für alle Personen i, so erfüllen die Zuweisungen a_{ij} und der Preisvektor $p = (p1, ..., pn)$ den komplementären Schlupf. Dieser liefert eine Verknüpfung zum dualen Problem. Letzteres besteht nach [6] in der Minimierung der Kostenfunktion

$$\sum_{i \in V} q_i(p) + \sum_{j \in V} p_j. \tag{15}$$

Bertsekas zeigt in [6], dass ein Preisvektor p, der den komplementäre Schlupf einhält, eine optimale Lösung des dualen Problems (15) liefert. Es wird zudem eine andere Darstellung der Dualität des Minimalkostenproblems über den Fluss geliefert, in der die Preise als Lagrangemultiplikatoren für die Erhaltung des Flusses auftreten [6]. Die Lagrangefunktion ist dabei durch

$$\begin{aligned} L(x,p) &= \sum_{(i,j) \in E} a_{ij} x_{ij} + \sum_{i \in V} \left(s_i - \sum_{\{j|(i,j) \in V\}} x_{ij} + \sum_{\{j|(j,i) \in V\}} x_{ji} \right) p_i \\ &= \sum_{(i,j) \in E} \left(a_{ij} + p_j - p_i\right) x_{ij} + \sum_{i \in V} s_i p_i \end{aligned} \tag{16}$$

gegeben. Das duale Problem ergibt sich nun als

$$\max_p q(p) = \max_p \left(\min_x \Big\{ L(x,p) | b_{ij} \leq x_{ij} \leq c_{ij} \Big\} \right). \tag{17}$$

Ein letzter Lösungsansatz bietet eine Interpretation der dualen Kostenverbesserung als „Auktion“. Hier werden zusätzliche Eigenschaften miteinbezogen, wie beispielsweise die Tatsache, dass jedes Gebot den vorherigen Preis übersteigen muss.
Auf Basis dieser drei angedeuteten Verfahren stellt Bertsekas verschiedene Algorithmen vor, wie beispielsweise eine preisbasierte Suche von augmentierenden Pfaden als Ergänzung der Methode von Ford und Fulkerson, die auch in nichtlinearen Netzwerken anwendbar sind [6].

2.6 Topologische Graph-Cuts

Eine wesentlich generalisiertere Version des Maximalflussproblems ist durch Krishnan und Ghrist in [25] und [38] gegeben. Die Autoren interpretieren das Problem in einem algebraisch-topologischen Raum. Dabei werden die Kapazitätsbeschränkungen als Garben kodiert, welche wiederum lokale Daten beschreiben [38]. Die Werte des Flusses beziehungsweise des Schnittes werden über (Ko-)Homologien dargestellt. Zudem konnte gezeigt werden, dass die Fluss-Schnitt-Dualität über eine topologische Poincaré-Dualität (siehe [41]) auf den genannten Garben gegeben ist [25]. Auf eine detaillierte Erläuterung der vorgestellten Konzepte wird an dieser Stelle verzichtet, da sie den Rahmen dieser Arbeit sprengen würde.

Schließlich zeigen Ghrist und Krishnan in [25] Anwendungen der topologischen Graph-Cut-Verfahren auf Flussprobleme mit multiplen Forderungen an den Fluss (*engl.: multicommodity flows*), Flüssen mit mehreren Quellen und Senken sowie Problemen mit logischen Flusswerten.

2.7 Graph-Cut-Verfahren auf der GPU

Vor allem in der Computer Vision oder Bildverarbeitung wird häufig mit großen Daten gearbeitet, so dass eine schnelle Ausführung der Algorithmen essentiell ist. Ansätze zur Reduktion der Rechenkomplexität wurden in den bisher beschriebenen Verfahren allesamt auf der CPU (Central Processing Unit) ausgeführt. Innerhalb der letzten Jahre hat sich jedoch die GPU (Graphics Processing Unit) als schneller Zusatzprozessor etabliert. Ursprünglich war dieser Graphikprozessor vor allem auf die Berechnung und Verarbeitung von rechenintensiven Aufgaben der Computergrafik zur Entlastung der CPU spezialisiert [42]. Durch die hohe Rechenleistung, welche viele Algorithmen mittlerweile fordern, reicht die Programmierung auf der CPU häufig nicht aus, um eine Ausführung in Echtzeit zu ermöglichen. Heutzutage werden daher auch allgemeinere Probleme auf der GPU gelöst. Dies ist bekannt als *General Purpose Programming on Graphics Processing Units (GPGPU)* [50].

Im Gegensatz zu der CPU, welche nur wenig Kerne besitzt (am verbreitetsten sind CPUs mit acht Kernen), verfügt die GPU über einen hochgradig parallelen Aufbau mit tausenden Kernen. Eine Programmiertechnik, die von dem Graphikprozessorentwickler Nvidia entwickelt wurde, stellt *CUDA (Compute Unified Device Architecture)* dar. Mithilfe eines Compilers wird ausführbarer Programmcode für die GPU erzeugt [50]. Die im Folgenden vorgestellten Verfahren wurden mithilfe von CUDA implementiert.

Im Jahr 2005 lagerten Dixit et al. [21] einige der Operationen eines Preflow-Push-Verfahrens auf die GPU aus. Dazu gehört beispielsweise die Entscheidung, ob eine *Push*- oder eine *Relabel*-Operationen durchgeführt werden muss, welche für alle Pixel parallel ausgeführt werden kann. Im Vergleich zu einer Version, die allein auf der CPU implementiert wurde, konnte die Rechenzeit jedoch nur unwesentlich verringert werden. Zwei Jahre spä-

ter entwickelten Hussein et al. [30] eine Implementierung von Graph-Cut-Verfahren auf der GPU. Wie auch Dixit et al. [21] arbeiteten sie dabei mit dem Preflow-Push-Verfahren, wandelten es aber insofern ab, dass die Parallelität der GPU besser ausgenutzt werden kann. Für eine parallele *Relabel*-Operation verwenden sie eine effiziente Breitensuche, um den Knoten des Graphen optimale Werte der Höhenfunktion zuweisen zu können. Ein Problem entsteht durch die Unmöglichkeit eines gleichzeitigen Ab- und Zuflusses in einem Knoten. Die Lösung stellt eine Aufteilung der *Push*-Operation in zwei Phasen dar. In der ersten *Push*-Phase wird kein Überschuss der Knoten aktualisiert, sondern das Ausmaß an Fluss gespeichert, welches in verschiedene Richtungen abfließen kann. Anschließend lesen alle Knoten eines bestimmten Tiefenlevels den Fluss, der zu ihnen fließt, aus dem zuvor gespeicherten Werten aus. Dies wird als *Pull*-Phase bezeichnet. Insgesamt kann das Verfahren eine Beschleunigung der Laufzeit im Vergleich zu schnellen Algorithmen auf der CPU um einen Faktor zwischen 1.7 und 4.5 liefern.

Im darauf folgenden Jahr ermöglichten Vineet und Narayanan [50] eine weitere Beschleunigung auf Basis des Verfahrens von Hussein et al. [30]. Ebenso wie ihre Vorgänger unterteilen sie die *Push*-Operation in zwei Phasen, wobei in der ersten lokalen Operation Fluss von jedem Knoten zu seinen Nachbarn abfließt und dabei den eigenen Überschuss verringert. Gespeichert wird dies ebenfalls in einem globalen Speicher, von dem ausgehend die zweite Operation die Informationen einliest und den Überschuss des Netzwerkes aktualisiert. Im Gegensatz zu Hussein et al. [30] teilen Vineet und Narayanan [50] auch die *Relabel*-Operation in zwei Schritte auf. In einer *lokalen Relabel-Operation* werden die Label aller Nachbarn eines Knotens aus einem globalen Speicher eingelesen und die neuen Label wieder hineingeschrieben. Dabei wird die Höhe eines Knotens um eine Einheit gegenüber des Minimums seiner Nachbarn erhöht. Die *globale Relabel-Operation* bestimmt, ähnlich zum Verfahren von Hussein et al. [30], mithilfe einer Breitensuche die wahren Distanzen zu der Senke. Die Aktualisierung ist dabei erneut ein äußerst paralleler Schritt. Insgesamt ergibt sich das Verfahren wie in Algorithmus 8 beschrieben. Im Vergleich zu dem Ansatz von Hussein et al. [30] kann eine weitere Beschleunigung um den Faktor 8 bis 10 erreicht werden. Dies kommt insbesondere durch die Aufteilung der *Relabel*-Operation zustande.

Algorithmus 8 Schnelle Graph-Cut-Verfahren auf der GPU nach [50]

1: Berechne die Kantengewichte aus dem Eingabebild
2: Führe für alle Knoten erst die *Push*-, anschließend die *Pull*-Operation aus
3: Wiederhole Schritt 2 m-mal
4: Führe für alle Knoten die *lokale Relabel*-Operation aus
5: Wiederhole die Schritte 2-4 k-mal
6: Wende die *globale Relabel*-Operation auf alle Knoten an
7: Wiederhole die Schritte 2-6 bis zur Konvergenz

2.8 Graph-Cut-Verfahren in der Bildsegmentierung

Graph-Cut-Verfahren finden in vielen Bereichen Anwendung. Beispiele sind Stereo-Probleme, das Entrauschen von Bildern und die Segmentierung. Im Rahmen der Computer Vision und Bildverarbeitung wurden sie erstmals durch die Arbeit von Greig et al. [27] bekannt, welche Maximalflussprobleme im Rahmen des Glättens verrauschter Bilder einsetzten. Später zeigten Boykov et al. [13, 14] die vielseitigen und effizienten Anwendungen der Graph-Cut-Verfahren in der Computer Vision. Anders als bei grundlegenden graphentheoretischen Ansätzen ist das Gitter des Graphen im Zusammenhang mit Bilddaten unveränderlich, da es durch die gegebenen Bilder festgelegt ist.

Üblicherweise werden Graph-Cut-Verfahren für automatische Segmentierungen eingesetzt. Die Ergebnisse können jedoch verbessert werden, indem eine interaktive Initialisierung von Vorder- und Hintergrund des Bildes durch die Benutzer vorgegeben wird, wie beispielsweise in [12]. Anschließend wird das Optimierungsproblem mittels gängiger Graph-Cut-Verfahren gelöst. Eine Reduktion der Interaktionen liefert beispielsweise das *GrabCut-Verfahren* von [44]. Auf Basis einer harten Segmentierung mit keinerlei Transparenz folgt eine Mattierung der Konturen. Der erste Schritt wird dabei mithilfe einer iterativen Energieminimierung durch Graph-Cut-Verfahren berechnet. Dabei wird die Verteilung der Pixelintensitäten in Vorder- und Hintergrund über GMM-Schätzer (Generalized Method of Moments) angegeben. Der iterative Ansatz führt zu einer Reduktion des Benutzeraufwandes [44]. Zusätzlich kann die Initialisierung der Segmentierung durch die Benutzer deutlich einfacher sein als bei dem interaktiven Graph-Cut-Verfahren von [12].

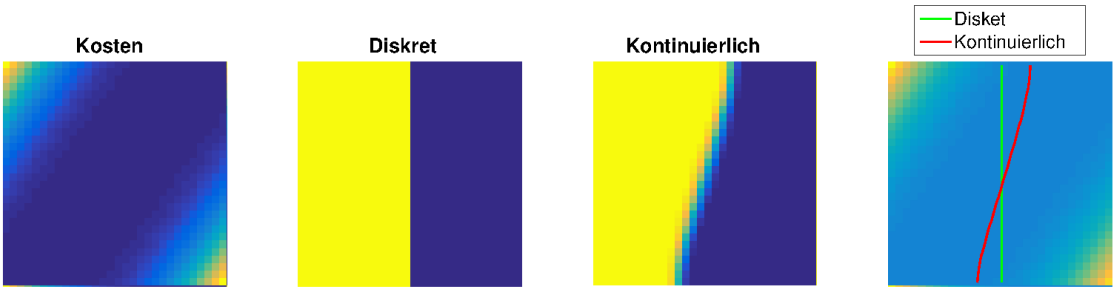


Abbildung 5. Segmentierung mittels zweier unterschiedlicher Graph-Cut-Ansätze. Links sind die gewählten Kosten zu sehen, wobei hohe Gewichte in gelb und niedrige in blau dargestellt sind. Es folgen eine Segmentierung mittels eines diskreten Graph-Cut-Verfahrens und mithilfe einer kontinuierlichen Methode. Rechts sind beide Schnitte mit den gewählten Kosten überlagert. Das diskrete Verfahren bevorzugt einen achsenparallelen Schnitt, wohingegen die kontinuierliche Methode eine bessere Anpassung an das Problem zeigt.

In der Anwendung haben diskrete Graph-Cut-Verfahren den Vorteil, dass sie Segmentierungsprobleme sehr schnell lösen können. Die Darstellung des Bildes als Graph legt jedoch dessen geometrischen Aufbau von Beginn an fest: Die Schnittkanten können nur entlang der Pixelgrenzen verlaufen. Dadurch, dass ein Bildpunkt mit seinen vier Nach-

barn verbunden wird, entsteht zudem eine Anisotropie, durch die bestimmte Richtungen präferiert werden. Dies resultiert in Diskretisierungsartefakten, die in der Segmentierung sichtbar werden. Ein gutes Beispiel ist in Abbildung 5 zu sehen. Das linke Bild zeigt die Wahl der Kosten, wobei hohe Kosten in gelb und niedrige in blau dargestellt sind. Ein Graph-Cut wurde hier mithilfe des Verfahrens von Boykov und Kolmogorov [11] durchgeführt. Die Segmentierung bevorzugt jedoch einen achsenparallelen Schnitt, obwohl ein schrägerer Schnitt bei Betrachtung der Kosten passender erscheint.

Eine Möglichkeit zur Vermeidung von Artefakten ist die Erweiterung der diskreten Verfahren auf kontinuierliche Graph-Cut-Methoden, welche in Kapitel 3 und 4 vorgestellt werden. Die rechte Segmentierung in Abbildung 5 wurde mithilfe eines kontinuierlichen Verfahrens erstellt und stellt eine passendere Approximation eines schrägen Schnittes dar.

Kapitel 3: Kontinuierliche Maximalfluss-Algorithmen

Graph-basierte Ansätze werden heutzutage in vielen Anwedungen der Bildverarbeitung und Computer Vision genutzt [13, 14, 27]. Um das in Abschnitt 2.3 beschriebene Problem der Diskretisierungsartefakte zu umgehen, wurden diese Verfahren im Laufe der letzten Jahre auf kontinuierliche Räume erweitert. Anstelle einer graph-basierten Energieminimierung wird mit einer kontinuierlichen Darstellung des Schnitts gearbeitet, um Anisotropien (Richtungsabhängigkeiten) zu vermeiden. Eine Diskretisierung folgt im Anschluss nur zu Zwecken des numerischen Lösens in Einklang mit der kontinuierlichen Formulierung. Da eine direkte Übertragung des diskreten Problems nicht möglich ist, müssen beispielsweise der Fluss und die Kapazitäten auf andere Weise interpretiert werden:

Der Fluss ist hier definiert als eine Funktion $\vec{F} : \Omega \to \mathbb{R}^2$ über ein kontinuierliches Gebiet Ω. Die Flusserhaltungsbedingung wird über die Divergenz definiert. Innerhalb eines Vektor- oder Strömungsfeldes beschreibt die Divergenz, wie sehr die Vektoren des Feldes in einer infinitesimal kleinen Umgebung auseinanderstreben. Zusätzlich gibt sie die Quelldichte

$$\text{div } \vec{F}(\vec{r}) \begin{cases} > 0, & \text{Quelle} \\ = 0, & \text{divergenzfrei} \\ < 0, & \text{Senke} \end{cases} \tag{18}$$

eines Strömungsfeldes $\vec{F}(\vec{r})$ an. Wie auch im diskreten Fall bezeichnen die Quelle und Senke den Anfangs- und Endpunkt des Flusses. Die Divergenzfreiheit entspricht der zuvor eingeführten Flusserhaltung [29].

Auch die Definition der Kapazitäten muss in der kontinuierlichen Darstellung angepasst werden. Sie sind hier in jedem Bildpunkt $x \in \Omega$ gegeben und nicht auf den Kanten eines Graphen dargestellt.

Im weiteren Verlauf dieses Kapitels wird ein grober Überblick über bisherige Ansätze dieses Gebietes gegeben. Einige dieser Verfahren unterscheiden sich allerdings von der in Kapitel 4 untersuchten kontinuierlichen Methode.

3.1 Kontinuierliche Maximalflussprobleme nach Strang [49]

Bereits 1983 untersuchte Strang in [49] und weiterführend in [48] Maximalfluss- und Minimalschnittprobleme im kontinuierlichem Raum. Analog zu einem diskreten Fluss auf Kanten werden hier kontinuierliche Flüsse durch ein Vektorfeld auf einem Gebiet Ω beschrieben. Zur Lösung des kontinuierlichen Problems soll eine Menge $S \subset \Omega$ gefunden werden. Der Rand ∂S dieser Menge beschreibt schließlich den maximalen Fluss. Die Flusserhaltung wird über die Divergenz definiert: Der Vektor $v = (v_1(x, y), v_2(x, y))$

besteht aus dem Wert und der Richtung des Flusses. Die Divergenz ist dann durch

$$\text{div } v = \frac{\partial v_1}{\partial x} + \frac{\partial v_2}{\partial y} = tF(x,y) \text{ in } \Omega \quad \text{(Quellen-/Senkenterm)} \tag{19}$$

gegeben. Auch im kontinuierlichen Raum ist der Fluss durch die Kapazitätsbedingung beschränkt. Diese wird hier nicht auf den Kanten des Graphen, sondern in jedem Punkt definiert:

$$|v(x,y)| = \sqrt{v_1^2 + v_2^2} \leq c(x,y) \text{ in } \Omega \quad \text{(Kapazitätsbedingung)}. \tag{20}$$

Das Maximalflussproblem ergibt sich als

$$\begin{aligned} &\max t, \text{ so dass} \\ &|v| \leq c \\ &\text{div } v = tF. \end{aligned} \tag{21}$$

Zusätzlich werden Randbedingungen gefordert, damit der Fluss nicht über den Rand hinaus fließen kann. Diese erste Darstellung eines kontinuierlicher Maximalflussproblems verallgemeinert jedoch das in Kapitel 4 vorgestellte Verfahren: Der Quellen-/Senkenterm F nimmt in dieser Masterarbeit nur für die Quelle und die Senke einen Wert ungleich von null an, um eine Divergenzfreiheit zu erfüllen.

3.2 Berechnung globaler minimaler Oberflächen durch Maximalfluss-probleme

Weiterführend präsentierten Appleton und Talbot [3] im Jahr 2006 einen Algorithmus zur Berechnung globaler minimaler Kurven und Oberflächen in Riemannräumen. Dabei wird ein minimaler Schnitt durch eine geschlossene und glatte Oberfläche S beschrieben, die Quelle und Senke voneinander trennt und dabei die gewichtete Länge $E(S)$ von S bezüglich der Metrik g mit

$$E(S) = \oint_S g(S) dS \tag{22}$$

minimiert. Der Fluss $\vec{F} : (\Omega, \mathbb{R}^+) \to \mathbb{R}^N$ ist hier ein Vektorfeld über ein kontinuierliches Gebiet mit zusätzlicher Zeitvariable τ. Um die Energie (22) zu minimieren, wird ein nichtlineares System partieller Differentialgleichungen gelöst. Darüber hinaus beschreibt $P : (\Omega, \mathbb{R}^+) \to \mathbb{R}$ ein Potentialfeld, das sich über Zeit entwickelt und überschüssigen Fluss (vergleichbar mit dem Überschuss) und dessen Weiterleitung darstellt:

$$\frac{\partial P}{\partial \tau} = -\nabla \cdot \vec{F} \tag{23}$$

$$\frac{\partial \vec{F}}{\partial \tau} = -\nabla P, \tag{24}$$

$$\text{so dass } |\vec{F}| \leq g. \tag{25}$$

Gleichung (23) relaxiert die Bedingung der Flusserhaltung für die Analogie zum Präfluss aus Abschnitt 2.3.2. Wie in [48] wird die Flusserhaltung über die Divergenz beschrieben,

welche wie in Gleichung (24) als negativer transponierter Gradient umgeformt werden kann. Die Gleichung modifiziert den Fluss so, dass die Kapazitätsbedingung aus (25) für alle Zeiten eingehalten wird. Sie fordert die Beschränkung des Flusses durch die Kapazitätsfunktion g. Es wird schließlich gezeigt, dass das Potentialfeld P gegen eine binäre Funktion konvergiert, welche die Quelle und die Senke voneinander trennt [3].

3.3 Kontinuierliche Maximalflussprobleme von Yuan et al. [53]

Im Jahr 2010 stellten Yuan et al. [53] einen weiteren kontinuierlichen Ansatz vor, den sie in den folgenden Jahren erweiterten [5, 54]. Anders als das in Kapitel 4 untersuchte kontinuierliche Verfahren werden die Quelle und Senke hier nach dem regionenbasierten Ansatz definiert. Gesondert zu den üblichen Bildpunkten werden sie außerhalb des Bildes platziert und mit allen anderen Punkten verbunden.

Zusätzlich unterteilen Yuan et al. den Fluss in einen räumlichen Fluss durch den Punkt x als $f(x)$, einen Fluss von der Quelle zum Punkt x als $f_s(x)$ und einen Fluss $f_t(x)$ von x zur Senke. Durch diese Aufteilung kann die Kapazitätsbedingung verfeinert werden:

$$|f(x)| \leq C(x) \qquad \forall x \in \Omega \tag{26}$$
$$f_s(x) \leq C_s(x) \qquad \forall x \in \Omega \tag{27}$$
$$f_t(x) \leq C_t(x) \qquad \forall x \in \Omega. \tag{28}$$

Die Kapazitäten sind hier auf den Bildpunkten definiert. Wie in den bisher vorgestellten kontinuierlichen Ansätzen wird die Divergenzfreiheit analog zur Flusserhaltung verwendet. Anders als in den Vorgängermodellen wird sie hier aufgrund der Aufteilung des Flusses leicht abgewandelt:

$$\mathrm{div} f(x) - f_s(x) + f_t(x) = 0 \text{ für fast alle } x \in \Omega. \tag{29}$$

Analog zum diskreten Fall präsentieren Yuan et al. [53] sowohl eine primale als auch eine duale Formulierung der Energieminimierung:

$$\sup_{f_s, f_t, f:\Omega \to \mathbb{R}^n} \left\{ P(f_s, f_t, f) := \int_\Omega f_s(x)\, \mathrm{d}x \right\} \text{ (primal)} \tag{30}$$

$$\min_{\lambda(x) \in [0,1]} \left\{ D(\lambda) = \int_\Omega (1 - \lambda(x)) C_s(x) + \lambda(x) C_t(x) dx + C(x) |\nabla \lambda(x)|\, \mathrm{d}x \right\} \text{ (dual)} \tag{31}$$

Dabei entspricht (30) dem Maximalfluss- und (31) dem Minimalschnittproblem.

3.4 Residuale Fast-Marching-Methode

Schließlich untersuchte Cremer [19] im Jahr 2014 einen kontinuierlichen Ansatz, basierend auf dem Verfahren von Ford und Fulkerson. Das Graph-Cut-Problem unterteilt das kontinuierliche Gebiet $\Omega \subseteq \mathbb{R}^2$ so, dass

$$\Omega = \Omega_1 \cup \Omega_2, \quad \Omega_1 \cap \Omega_2 = \emptyset. \tag{32}$$

Dies entspricht einer binären Segmentierung. Zur Lösung des Problems wird eine Energieminimierung der Funktion $u : \Omega \to \{0, 1\}$ mit $\Omega_1 = u^{-1}(1)$, $\Omega_2 = u^{-1}(0)$ gelöst. Um eine Segmentierung zu erhalten, wird eine Einteilung der Bildpunkte der Quelle x_s und Senke x_t in unterschiedliche Segmente durch die Forderungen $u_s := u(x_s) = 1$, $u_t := u(x_t) = 0$ erzwungen. Daraus ergibt sich das Energieminimierungsproblem

$$\min_{\substack{u:\Omega\to\{0,1\},\\ u(x_s)=0,\\ u(x_t)=1}} \int_\Omega c(x) \cdot \|\nabla u(x)\|_p \, \mathrm{d}x. \tag{33}$$

Die Bezeichnung ∇u wird hier im Sinne einer Distribution aufgefasst. Andernfalls könnte kein Gradient für die Funktion $u : \Omega \to \{0, 1\}$ an den Sprungstellen definiert werden. Zusätzlich sind, anders als bei den diskreten Verfahren, die Kapazitäten auf den Bildpunkten definiert und nicht auf den Kanten eines Graphen. Die von Cremer in [19] untersuchte *residuale Fast-Marching-Methode* löst die duale Formulierung des Problems (33) unter Verwendung der Fast-Marching-Methode. Diese bezeichnet ein numerisches Verfahren, das die Eikonalgleichung

$$\begin{aligned} F(x) \cdot |\nabla \phi(x)| &= 1 \\ \phi(s) &= 0 \end{aligned} \tag{34}$$

mit $F, \phi : \Omega \to \mathbb{R}$ löst. Sie beschreibt ein Distanzmaß ausgehend von einem Startpunkt s [47]. Anwendung findet die Lösungsmethode unter anderem in der Suche nach kürzesten Pfaden durch einen Graphen.

Um praktisch mit dem Energieminimierungsproblem (33) arbeiten zu können, wird im Folgenden die Diskretisierung

$$\min_{u\in\mathbb{R}^n} \sum_{i=1}^{n} c_i \cdot \|(Du)_i\|_p, \quad \text{so dass } u_s = 1, u_t = 0 \tag{35}$$

verwendet. Zusätzlich stellt $D \in \mathbb{R}^{2n\times n}$ eine Diskretisierung des Gradienten dar. Das Gebiet Ω ist gegeben durch die Menge aller Bildpunkte $\{1, \ldots, n\}$. Die duale Formulierung als Maximalflussproblem ergibt sich nun als

$$\max_f (-D^T f)_s, \text{ so dass} \tag{36}$$

$$|f_i|_q \le c_i \quad \forall i \in \{1, \ldots, n\}, \tag{37}$$

$$(-D^T f)_i = 0 \quad \forall i \in \{1, \ldots, n\} \backslash \{s, t\}. \tag{38}$$

Der negative transponierte Gradient $-D^T$ kann analog zum gauß'schen Integralsatz [24] als Divergenz aufgefasst werden. Der Fluss f wie auch die Kapazitäten sind auf den Bildpunkten definiert. Die Aufrechterhaltung der Dualität fordert zudem die Bedingung $\frac{1}{p} = \frac{1}{q} = 1$ (siehe dazu auch [19]). Im Weiteren wird $p = q = 2$ angenommen. Wie auch in den anderen kontinuierlichen Verfahren fordert die duale Formulierung eine Kapazitätsbedingung und eine Divergenzfreiheit außer in Quelle und Senke (siehe die Gleichungen (37)-(38)).

Die residuale Fast-Marching-Methode berechnet eine Aufstiegsrichtung für das Problem (36)-(38). Für deren Definition werden in diesem Zusammenhang die Begriffe des *Flusses* und des *Pseudoflusses* im kontinuierlichen Raum erläutert:

Definition 11 (Pseudofluss und Fluss nach [19])**.** Sei $c \in \mathbb{R}^n$ ein Vektor mit Knotenkapazitäten und $D \in \mathbb{R}^{2n \times n}$ eine Diskretisierung des Gradienten. Dann heißt $f \in \mathbb{R}^{n \times 2}$ ein *Pseudofluss*, wenn die Bedingung (37) erfüllt ist. Ein Pseudofluss f heißt *Fluss*, wenn zusätzlich die Bedingung (38) erfüllt ist.

Definition 12 (Aufstiegsrichtung nach [19])**.** Sei $D \in \mathbb{R}^{2n \times n}$ eine Diskretisierung des Gradienten. Dann heißt $\Delta f \in \mathbb{R}^{n \times 2}$ *Aufstiegsrichtung* von f bezüglich D, wenn

1. Δf in Bezug auf den Gradienten D für alle Punkte außer der Quelle und Senke divergenzfrei ist, das heißt $(-D^T \Delta f)_i = 0$ für $i \in \Omega \backslash \{s, t\}$ und
2. $f + \lambda \Delta f$ ein Pseudofluss für ein $\lambda > 0$ ist.

Durch die Berechnung der Aufstiegsrichtung findet die Methode einen augmentierenden Pfad durch den Graphen. Analog zum Verfahren von Ford und Fulkerson [23] wird der Fluss entlang dieser Pfade iterativ erhöht. Die vollständige residuale Fast-Marching-Methode zeigt Algorithmus 9. Für die angemessene Wahl des Gradienten siehe [19].

Algorithmus 9 Residual-Fast-Marching-Methode [19]

Eingabe: Graph G, Quelle und Senke $s, t \in \Omega = \{1, \ldots, n\}$, Knotenkapazitäten $c \in \mathbb{R}^n$
Ausgabe: s-t-Schnitt $S \subseteq \{1, \ldots, n\}$

1: Initialisiere den Pseudofluss $f := 0$
2: **Wiederhole**
3: Berechne Δf und D, so dass Δf Aufstiegsrichtung von v in Bezug auf D und c
4: $f := f + \Delta f$
5: Aktualisiere Kapazitätsbeschränkungen c
6: **bis** keine Aufstiegsrichtung mehr gefunden werden kann
7: **Rückgabe:** Menge S von allen von s aus erreichbaren Punkten im Residualgraphen

Die Berechnung der Aufstiegsrichtung in Schritt 3 des Algorithmus' 9 folgt über die Fast-Marching-Methode. Jedoch wird eine modifizierte Eikonalgleichung gelöst, um die

Kapazitätsbedingung weiterhin einhalten zu können:

$$|(\nabla\phi)_i|_2 = 1 \quad \forall i \in \{1,\dots,n\}\backslash\{s\}, \tag{39}$$

$$(\nabla\phi)_i \in \text{int } (N_{c_i|\cdot|_2}(v_i))^* \quad \forall i \in \{1,\dots,n\}, \tag{40}$$

$$\phi_s = 0. \tag{41}$$

Dabei bezeichnet $N_{c_i|\cdot|_2}$ den Normalkegel bezüglich der Menge $\{c_i|\cdot|_2\}$. Der Stern * visualisiert den dualen Kegel.

Definition 13 (Kegel)**.** Sei $C \subseteq \mathbb{R}^n$. C heißt *Kegel*, wenn $0 \in C$, $\lambda x \in C$ für alle $x \in C, \lambda \geq 0$. Ist C eine nichtleere, konvexe Menge, dann heißt N_C für $x \in C$ mit

$$N_C(x) = \{s : \langle s, y - x\rangle \leq 0 \text{ für alle } y \in C\} \tag{42}$$

der *Normalkegel von* C. Sei V ein Hilbertraum und C ein Kegel in V. Der *duale Kegel von* C ist definiert als

$$C^* = \{y \in V | \forall x \in C : \langle x, y\rangle \geq 0\}. \tag{43}$$

Die Gleichung (40) stellt nun zusätzlich zur gewöhnlichen Eikonalgleichung sicher, dass die Kapazitätsbedingung $|f_i| \leq c_i \ \forall i \in \Omega$ für den Fluss $f + \lambda\Delta$ gilt, wenn λ klein genug ist. Somit wird bei der Berechnung des Schritts sichergestellt, dass eine zulässige Aufstiegsrichtung gesucht wird.

Auch wenn das Verfahren gute Ergebnisse erzeugen kann, sind bislang keine Aussagen über seine Konvergenz bekannt. Ein großer Nachteil besteht darin, dass sich in jedem Schritt die Diskretisierung D des Gradienten ändert, wodurch keine einheitliche diskretisierte Energie angegeben werden kann, welche vom Verfahren minimiert wird. Verschiedene Möglichkeiten, um die resultierenden Probleme zu umgehen, sind in [19] beschrieben.

Weitere mögliche Ansätze für kontinuierliche Graph-Cut-Verfahren wurden beispielsweise von Chambolle et al. in [16] und Couprie et al. [18] vorgestellt. Erstere verwenden konvexe Methoden, um die nichtlinearen partiellen Differentialgleichungen (23)-(25) zu lösen. Couprie et al. [18] lösen das Problem mithilfe eines primal-dualen Interior-Point-Verfahrens. Im Weiteren wird nun ein weiterer von Cremer [19] entwickelte Ansatz und dessen Probleme sowie Verbesserungen vorgestellt.

Kapitel 4: Methode des erweiterten Residualgraphen

Die residuale Fast-Marching-Methode beschreibt sowohl die Kapazitäten als auch den Fluss auf den Pixeln eines Bildes. Im Jahr 2011 stellten Couprie et al. in [18] eine Kombination vor, bei der die Kapazitäten auf den Knoten definiert werden (siehe Abbildung 6), der Fluss jedoch auf den Kanten des Graphen. Die Zulässigkeit eines Knotens ist somit gegeben, wenn dessen Kapazität größer oder gleich der Norm des Flusses seiner vier benachbarten Kanten ist. Diese Idee wird auch in der *Methode des erweiterten Residualgraphen (ERG)* von Cremer in [19] aufgegriffen. Anders als Couprie et al. in [18] wird dabei ein kombinatorischer Lösungsansatz verwendet.

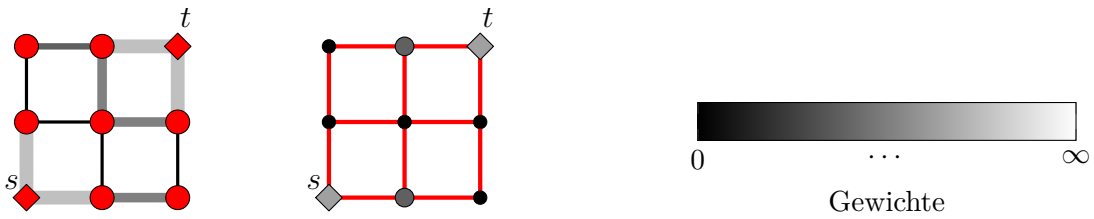


Abbildung 6. Vergleich der Definition der Kapazitäten auf den Kanten (linker Graph) und auf Knoten (rechter Graph). Die Quelle und die Senke sind als Rauten abgebildet. Die Skalierung der Gewichte ist rechts dargestellt. Kleine Kosten sind dunkel als dünne Kanten (links) beziehungsweise Knoten (mittig) darstellt, hohe Kosten analog hell und größer. Werden die Kosten auf den Kanten definiert, wird ein Schnitt durch die dünnsten schwarzen Kanten berechnet. Die Definition von Knotenkosten resultiert in einem Schnitt, der durch die kleinen schwarzen Knoten gegeben ist. Die Abbildung wurde erstellt nach [18, Seite 11].

4.1 Verfahren nach Cremer [19]

Um ein geeignetes Optimierungsproblem aufstellen zu können, müssen das primale Problem (33) und dessen duale Formulierung (36)-(38) angepasst werden. Die Bedingungen der Divergenzfreiheit und Kapazitätsbeschränkung können im Wesentlichen übernommen werden.

Für jeden Knoten wird der Fluss seiner vier Nachbarkanten betrachtet. Die Flüsse in vertikaler Richtung werden mit dem hochgestellten Index 1 versehen, in horizontaler Richtung mit dem Wert 2. Zusätzlich werden Flüsse auf Kanten links von dem aktuellen Knoten $v \in V$ oder oberhalb des Knotens mit einem „$-$“ versehen, die übrigen Flüsse mit einem „$+$“. Um anzuzeigen, für welchen Knoten die Indizierung vorgenommen wird, erhalten alle vier Flüsse den tiefgestellten Index $v \in V$ (siehe Abbildung 7). Der Gesamtfluss $f \in \mathbb{R}^{4n}$ mit $f = (f_v)_{v=1}^n$ wird im Folgenden durch $f_v = (f_v^{1+}, f_v^{1-}, f_v^{2+}, f_v^{2-})$ für alle Knoten $v \in V$ dargestellt.

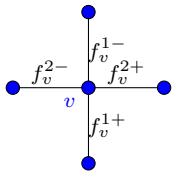


Abbildung 7. Darstellung des Flusses $f_v = (f_v^{1+}, f_v^{1-}, f_v^{2+}, f_v^{2-})$ eines Knotens v.

Diese redundante Betrachtung des Flusses fordert allerdings eine weitere Bedingung, welche die Kompatibilität des Flusses für benachbarte Kanten gewährleistet. Daher wird zusätzlich zu der in [19] eingeführten Darstellung im Rahmen dieser Masterarbeit eine Hilfsmatrix $A \in \{-1, 0, 1\}^{(n_1-1)n_2+n_1(n_2-1)\times 4n}$ für einen Graphen der Größe $n_1 \times n_2$ mit $n = n_1 \cdot n_2$ eingeführt. Sie sorgt dafür, dass der Fluss auf einer Verbindungskante zweier Knoten denselben Wert besitzt, zum Beispiel für $f_{v_{i,j}}^{2+} = f_{v_{i,j+1}}^{2-}$. Somit ergibt sich das Maximierungsproblem nach [19] mit der beschriebenen Erweiterung zunächst als

$$\max_{f:E\to\mathbb{R}} \text{div } f_s, \quad \text{so dass} \tag{44}$$

$$f_v \text{ zulässig} \quad \forall v \in V\backslash\{s,t\}, \tag{45}$$

$$\text{div } f_v = 0 \quad \forall v \in V\backslash\{s,t\}, \tag{46}$$

$$Af = 0. \tag{47}$$

Dabei sind $s = (s_1, s_2)$ und $t = (t_1, t_2) \in V$. Aufgrund dieser Darstellung des Flusses wird der Divergenzoperator hier wie folgt definiert [19]:

$$\text{div } f_v = -f_v^{1-} - f_v^{2-} + f_v^{1+} + f_v^{2+}. \tag{48}$$

An dieser Stelle soll darauf hingewiesen werden, dass die Kapazitätsbedingung (45) weiterhin adaptiert werden muss und im Verlauf des Abschnittes berichtigt wird. Vorerst wird zur Verdeutlichung des Prinzips die Formulierung $\|f_v\| \leq c_v$ verwendet, wobei die Norm in jedem Punkt gegeben ist. Um auch das primale Problem mit diesen Bedingungen an den Fluss anzupassen, wird eine duale Variable $w \in \mathbb{R}^{(n_1-1)n_2+n_1(n_2-1)}$ eingeführt, um die Bedingung (47) einhalten zu können. Der Übersichtlichkeit halber wird $m := (n_1 - 1)n_2 + n_1(n_2 - 1)$ definiert. Zusätzlich wird die Variable $u \in \mathbb{R}^n$ betrachtet, mithilfe derer die Divergenzfreiheit für alle Knoten außer der Quelle und Senke dargestellt werden kann. Dabei entspricht u der Variable aus Abschnitt 3.4, für die gilt, dass $u(s) = u_s = 1$ und $u(t) = u_t = 0$. Mithilfe der Indikatorfunktion $\delta_C : \mathbb{R}^n \to \mathbb{R}, C \subseteq \mathbb{R}^n$ mit

$$\delta_C(x) = \begin{cases} 0 & x \in C \\ +\infty & x \notin C \end{cases} \tag{49}$$

kann die Kapazitätsbedingung in den zu maximierenden Term eingebaut werden. Der Übersichtlichkeit halber wird nun statt der Bezeichnung div$f$ die äquivalente Formulierung über den Gradienten $D^T f$ verwendet. Über folgende Umformungsschritte kann schließlich die primale Formulierung des Problems hergeleitet werden. Dabei wird die

Divergenzfreiheit $D^T f = 0$ für alle Knoten bis auf die Quelle und Senke betrachtet. Zudem werden die verwendeten Normen in jedem Punkt berechnet.

$$\sup_{f\in\mathbb{R}^{4n}} \left\{ \underbrace{(D^T f)_s}_{=\text{ div } f_s} + \underbrace{\sum_{v\in V} \delta_{\|f_v\|\le c_v}}_{\text{Kapazitätsbedingung}} + \inf_{\substack{w\in\mathbb{R}^m,\\ u\in\mathbb{R}^n}} \Big\{ \underbrace{\langle Af, w\rangle}_{\text{Kompatibilität}} + \underbrace{\langle D^T f, u\rangle}_{\text{Divergenzfreiheit}} \Big\} \right\},$$

so dass $u(s) = 1, u(t) = 0$

$$= \sup_{f\in\mathbb{R}^{4n}} \left\{ (D^T f)_s + \sum_{v\in V} \delta_{\|f_v\|\le c_v} + \inf_{\substack{w\in\mathbb{R}^m,\\ u\in\mathbb{R}^n,\\ u_s=1, u_t=0}} \Big\{ \langle Af, w\rangle + \langle Du, f\rangle \Big\} \right\}$$

Die Bedingungen $u_s = 1$ und $u_t = 0$ werden anschließend über die Indikatorfunktion in den Term eingebaut:

$$= \sup_{f\in\mathbb{R}^{4n}} \left\{ (D^T f)_s + \sum_{v\in V} \delta_{\|f_v\|\le c_v} + \inf_{\substack{w\in\mathbb{R}^m,\\ u\in\mathbb{R}^n}} \Big\{ \langle Af, w\rangle + \langle Du, f\rangle + \delta_{\{0\}}(u_t) + \delta_{\{0\}}(1-u_s) \Big\} \right\}$$

Diese beiden Bedingungen sorgen dafür, dass der zu maximierende Ausdruck $(D^T f)_s$ in die Optimierung $\inf_u \langle Du, f\rangle$ aufgenommen werden kann:

$$= \sup_{f\in\mathbb{R}^{4n}} \inf_{\substack{w\in\mathbb{R}^m,\\ u\in\mathbb{R}^n}} \left\{ \langle Du, f\rangle + \langle Af, w\rangle + \sum_{v\in V} \delta_{\|f_v\|\le c_v} + \delta_{\{0\}}(u_t) + \delta_{\{0\}}(1-u_s) \right\}$$

$$= \sup_{f\in\mathbb{R}^{4n}} \inf_{\substack{w\in\mathbb{R}^m,\\ u\in\mathbb{R}^n}} \left\{ \langle f, Du + A^T w\rangle + \sum_{v\in V} \delta_{\|f_v\|\le c_v} + \delta_{\{0\}}(u_t) + \delta_{\{0\}}(1-u_s) \right\}$$

An dieser Stelle wird starke Dualität (S.D.) gefordert. Wenn diese Bedingung eingehalten wird, sind die primale und duale Lösung des Problems äquivalent [8].

$$\begin{aligned} &\overset{\text{S.D.}}{=} \inf_{\substack{w\in\mathbb{R}^m,\\ u\in\mathbb{R}^n}} \sup_{f\in\mathbb{R}^{4n}} \left\{ \langle f, Du + A^T w\rangle + \sum_{v\in V} \delta_{\|f_v\|\le c_v} + \delta_{\{0\}}(u_t) + \delta_{\{0\}}(1-u_s) \right\} \\ &= \inf_{\substack{w\in\mathbb{R}^m,\\ u\in\mathbb{R}^n}} \left\{ \sum_{v=1}^{n} c_v \|(Du + A^T w)_v\| + \delta_{\{0\}}(u_t) + \delta_{\{0\}}(1-u_s) \right\}. \end{aligned} \tag{50}$$

Die Kapazitätsbedingung (45) wird im Folgenden über die Flüsse der vier benachbarten Kanten $e_1, e_2, e_3, e_4 \in E$ des Knotens $v \in V$ beschrieben. Der Einfachheit halber wird auf die Bezeichnungen der Nachbarflüsse als $f_v^{2-}, f_v^{1+}, f_v^{2+}$ und f_v^{1-} verzichtet. Wird ein s-t-Pfad P bestimmt, der durch den Knoten v läuft, so stehen die Kanten e_1 und e_2 für jene Kanten, die von P durch v verwendet werden. Die Kanten e_3 und e_4 bleiben durch den Pfad unbenutzt. Somit folgt die Kapazitätsbeschränkung

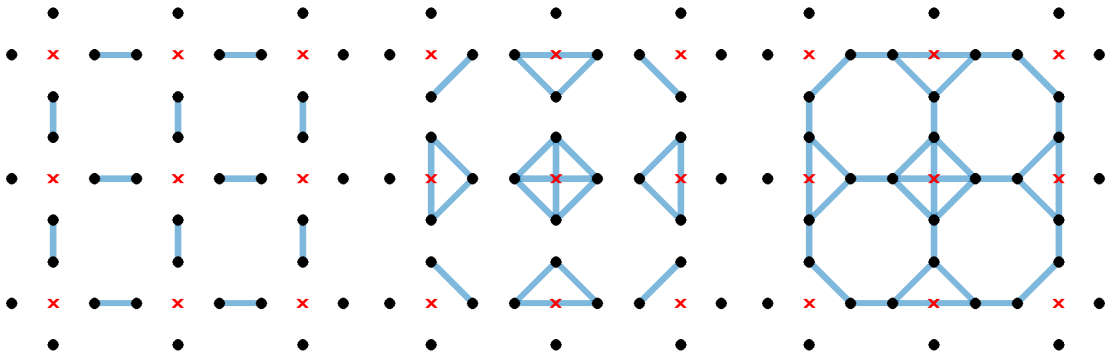
$$(f_v^{2-})^2 + (f_v^{1+})^2 + (f_v^{2+})^2 + (f_v^{1-})^2 = (f_{e_1})^2 + (f_{e_2})^2 + (f_{e_3})^2 + (f_{e_4})^2 \le c_v^2. \tag{51}$$

An seinem Pfad P durch den Graphen kann nun eine Aufstiegsrichtung für Gleichung (50) konstruiert werden, wenn zusätzlich zu f auch $f + \epsilon \cdot \delta_P$ weiterhin zulässig ist. Dabei bezeichnet δ_P eine Richtung entlang P und ϵ eine Schrittweite. Da der Pfad nur maximal zwei der benachbarten Kanten von v verwenden kann, um durch einen Knoten zu fließen, ergibt sich die Kapazitätsbedingung nach [19] als

$$(f_{e_1} + (-1)^{r_1}\epsilon)^2 + (f_{e_2} + (-1)^{r_2}\epsilon)^2 + (f_{e_3})^2 + (f_{e_4})^2 \leq c_v^2. \tag{52}$$

Die Vorzeichen r_1 und r_2 hängen hier von der Richtung des Pfades durch die jeweiligen Kanten ab. Für Flüsse, die in einen Knoten einfließen, sind r_1 beziehungsweise r_2 positiv, andernfalls negativ. Aufgrund der unterschiedlichen Form der Kapazitätsbedingung und des Flusses kann für diese Methode kein gewöhnlicher Residualgraph verwendet werden.

Der *erweiterte Residualgraph* wird nun so konstruiert, dass jeder Knoten v durch vier Unterknoten v_k mit $k \in \{1, 2, 3, 4\}$ ersetzt wird. Jeder Block aus Unterknoten wird wie der ursprüngliche Graph G durch Kanten mit seinen Nachbarn verbunden (Kanten zwischen unterschiedlichen Elternknoten, siehe Abbildung 8a). Wird die Kapazitätsbedingung (51) für einen Knoten v eingehalten, so werden auch die Unterknoten innerhalb eines Blockes verbunden (Kanten gleicher Elternknoten, siehe Abbildung 8b). Die Sättigung eines Knotens führt zur Entfernung der entsprechenden Kante des gleichen Elternknotens aus dem erweiterten Residualgraphen.



(a) Kanten zwischen unterschiedlichen Elternknoten. **(b)** Kanten gleicher Elternknoten. **(c)** Vollständiger erweiterter Residualgraph.

Abbildung 8. Aufbau des erweiterten Residualgraphen, bestehend aus Kanten zwischen gleichen und unterschiedlichen Elternknoten (blau). Rote Kreuze deuten die ursprünglichen Knoten an, welche im erweiterten Residualgraphen durch jeweils vier Knoten ersetzt werden (schwarze Kreise).

Die Überprüfung der Kapazitätsbedingung und das Finden eines augmentierenden Pfades folgt durch die Lösung der quadratischen Gleichung in (52) nach der Unbekannten ϵ, die eine Schrittweite des Flusses darstellt. Dabei werden alle zwölf Möglichkeiten betrachtet, einen Pfad durch einen Knoten zu konstruieren (siehe Abbildung 10) und für all diese die Existenz einer Lösung der Kapazitätsbedingung (52) geprüft. Ein möglicher

Pfad durch den Graphen kann nur dann als augmentierender Pfad genutzt werden, wenn die Gleichung für jeden Knoten innerhalb des Pfades eine Lösung findet, also in jedem Knoten ein $\epsilon > 0$ existiert. Nimmt die Schrittweite ϵ für einen Knoten den Wert null an, so wird dieser als gesättigt gespeichert.

Der Zusammenhang des erweiterten Residualgraphen G_f mit dem Ursprungsgraphen G wurde in [19] mithilfe von *Wegen* bewiesen. Diese bezeichnen Pfade durch den Graphen, bei denen ein Knoten mehrfach verwendet werden kann (siehe Definition 4).

Satz 2. Sei $G = (V, E)$ ein Graph mit Quelle und Senke $s, t \in V$. Dann besitzt G einen augmentierenden s-t-Weg, wenn ein s_1-t_1-Pfad im erweiterten Residualgraph G_f von G existiert.

Beweis. Siehe Abschnitt A.2 oder [19]. □

Das vollständige Verfahren beruht auf der Idee von Ford und Fulkerson [23]. Dabei wird der erweiterte Residualgraph G_f in jedem Schritt konstruiert und die Existenz eines augmentierenden Pfades mithilfe der Kapazitätsbedingung (52) für die Schrittweite ϵ geprüft. Entlang eines solchen Pfades wird der Fluss solange erhöht, bis in G_f kein Pfad mehr auffindbar ist (siehe Algorithmus 10).

Bislang blieb die Frage unbeantwortet, ob mithilfe des Verfahrens tatsächlich immer ein Minimierer der Energie (33) gefunden werden kann. Daran knüpft auch das Problem an, ob aus der Existenz einer Aufstiegsrichtung die Existenz eines s-t-Weges in G folgt.

Algorithmus 10 Methode des erweiterten Residualgraphen nach [19]

Eingabe: Quelle und Senke $s, t \in \{1, \ldots, n\}$, Knotenkapazitäten $c \in \mathbb{R}^n$
Ausgabe: Maximaler Fluss f
1: Initialisiere $f := 0$
2: **Wiederhole**
3: Konstruiere den erweiterten Residualgraphen G_f mithilfe von f
4: Berechne den kürzesten s_1-t_1-Pfad in G_f und extrahiere s-t-Weg P in G
5: Berechne ein maximales $\epsilon > 0$, so dass $f + \epsilon\delta_P$ zulässig ist
6: $f := f + \epsilon\delta_P$
7: **bis** kein s_1-t_1-Pfad in G_f existiert
8: **Rückgabe:** f

4.1.1 Vorarbeiten

Die bereits vorhandene Implementierung des ERG-Verfahrens verwendet sowohl MATLAB als auch C++. Dazu wurde ein Teil der Funktionen in C++ implementiert, die jedoch innerhalb von MATLAB aufgerufen und mittels `mex` (als Abkürzung für *MATLAB executable*) kompiliert werden können [2]. Dies bietet den Vorteil, dass die Effizienz durch die C++-Implementierung deutlich verbessert werden kann. Die im Folgenden beschriebene Implementierung wurde von Cremer auf der Basis von [19] angefertigt. Alle weiteren Lösungsvorschläge aus Abschnitt 4.2 wurden im Rahmen dieser Masterarbeit zu den vorhandenen Programmen hinzugefügt.

Die Konstruktion des erweiterten Residualgraphen in Schritt 3 von Algorithmus 10 wurde als `mex-Datei` in C++ programmiert. Ein wichtiges Ausgabeargument der Funktion ist die Adjazenzmatrix des erweiterten Residualgraphen, die angibt, welche Knoten durch Kanten miteinander verbunden werden. Für jeden Knoten $v \in V$ werden vier Unterknoten v_1, v_2, v_3 und v_4 zur Knotenmenge $V(G_f)$ des erweiterten Residualgraphen hinzugefügt. Die Nummerierung folgt gegen den Uhrzeigersinn, wobei v_1 rechts von v angeordnet wird (siehe Abbildung 9).

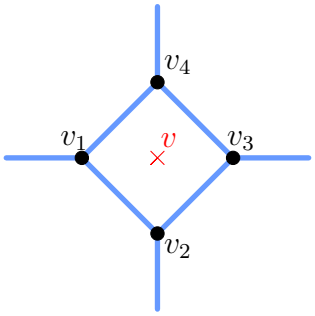


Abbildung 9. Darstellung der Anordnung der vier Unterknoten eines Knotens v.

Im ersten Schritt der Konstruktion von G_f werden die Kanten zwischen unterschiedlichen Elternknoten hinzugefügt. Für jede Kante $e = (v, w) \in E$ werden die Kanten e_1 und e_2 nach [19] zur Kantenmenge $E(G_f)$ hinzugefügt, wobei

$$\begin{aligned}
e_1 = (v_3, w_1), e_2 = (w_1, v_3), &\quad \text{wenn } w \text{ rechts neben } v \text{ liegt,} \\
e_1 = (v_1, w_3), e_2 = (w_3, v_1), &\quad \text{wenn } w \text{ links neben } v \text{ liegt,} \\
e_1 = (v_2, w_4), e_2 = (w_4, v_2), &\quad \text{wenn } w \text{ unterhalbe von } v \text{ liegt,} \\
e_1 = (v_4, w_2), e_2 = (w_2, v_4), &\quad \text{wenn } w \text{ oberhalb von } v \text{ liegt.}
\end{aligned} \tag{53}$$

Jede Verbindung zweier Knoten v_i und w_j im erweiterten Residualgraphen wird mit dem Wert 1 innerhalb des Eintrags a_{v_i,w_j} der Adjazenzmatrix $A = (a_{ij})_{i,j=1}^{4n}$ gespeichert.

Anschließend wird berechnet, welche Knoten (un-) gesättigt sind. Eine Kante innerhalb des gleichen Elternknotens wird nur dann zum erweiterten Residualgraphen G_f hinzugefügt, wenn der zugehörige Elternknoten nicht gesättigt ist. Bei einer 4er-Nachbarschaft im Graphen G existieren genau 12 Möglichkeiten, einen Pfad durch einen Knoten v zu konstruieren (siehe Abbildung 10).

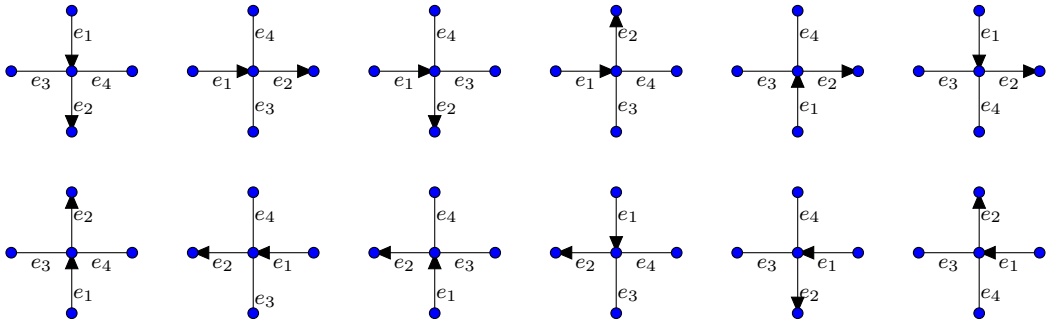


Abbildung 10. Darstellung aller 12 Möglichkeiten, einen Pfad durch einen Knoten zu konstruieren. Die Richtung des Flusses wird durch Pfeile angezeigt.

Für die weitere Vorgehensweise werden diese 12 Möglichkeiten nacheinander für jeden Knoten $v \in V$ betrachtet. Dabei bezeichnet e_1 die Kante, über die Fluss in den Knoten v einfließt, e_2 die Kante, über die Fluss von v ausfließt und e_3 und e_4 die unbenutzten Kanten (siehe Abbildung 10).

Für jeden Knoten und alle Flussmöglichkeiten wird nun eine Lösung der Gleichung (52) berechnet. Kann für eine der Möglichkeiten eine Schrittweite $\epsilon > \tau$ für eine kleine Toleranzgrenze $\tau \geq 0$ bestimmt werden, ist weiterer Fluss durch den aktuellen Knoten möglich. Es kann potentiell ein augmentierender Pfad durch diesen Knoten konstruiert werden. Die Kanten innerhalb dieses Elternknotens werden zum erweiterteten Residualgraphen hinzugefügt und in der Adjazenzmatrix von G_f gespeichert.

Zusätzlich werden die Werte möglicher positiver Schrittweiten ϵ in einer Matrix $\mathcal{E} \in \mathbb{R}^{n\times 4\times 4}$ gespeichert. Das Element $\mathcal{E}(v, \text{direction}_1, \text{direction}_2)$ gibt nun Auskunft darüber, um wie viel der Fluss im Knoten v von Richtung direction_1 nach direction_2 erhöht werden kann, ohne die Kapazitätsbeschränkung zu verletzen. Dabei sind die Richtungen direction_1 und direction_2 abhängig davon, welche der 12 Möglichkeiten aktuell betrachtet wird. Indes beschreibt direction_1 die Richtung des Flusses auf der Kante e_1 und direction_2 die Richtung des Flusses von e_2. Die Speicherung aller Schrittweiten ist nötig, um im Anschluss den maximal möglichen Wert einer Flusserhöhung durch G_f bestimmen zu können. Daher wird auch die Matrix $\mathcal{E}$ als Rückgabewert der C++-Datei definiert.

Die C++-Funktion zur Konstruktion der Adjazenzmatrix des erweiterten Residualgraphen wird im nächsten Schritt in MATLAB aufgerufen. Dazu wird innerhalb von G_f

der kürzeste Pfad von der Quelle zur Senke mithilfe des Verfahrens von Dijkstra [17] gesucht. Dabei werden die obersten Knoten s_4 und t_4 der Vierblöcke der Elternknoten s und t als Start- und Endpunkte für diese Suche verwendet. Das Verfahren terminiert, sobald kein Pfad mehr auffindbar ist.

Für jeden Pfad innerhalb des erweiterten Residualgraphen wird ein Weg im Ursprungsgraphen G rekonstruiert und die entsprechende Schrittweite ϵ aus der Matrix $\mathcal{E}$ ausgelesen. Diese ist die maximal mögliche Flusserhöhung entlang des berechneten Pfades durch G_f beziehungsweise des rekonstruierten Weges durch G. Der Fluss wird anschließend analog zu den vorgestellten diskreten Verfahren aus Kapitel 2 um diesen Wert ϵ entlang des Pfades erhöht.

Algorithmus 11 Bestimmung der (un-) gesättigten Knoten [19]

Eingabe: Fluss $f : E \to \mathbb{R}$
Ausgabe: Adjazenzmatrix A, Schrittweiten $\mathcal{E}$

1: **Für** alle Knoten $v \in V$
2: **Für** alle 12 Möglichkeiten *perm*, Fluss durch einen Knoten fließen zu lassen
3: Bestimme die Nachbarknoten $w_i, i = \{1, 2, 3, 4\}$ von v in der durch *perm* vorgegebenen Reihenfolge
4: Bestimme den Wert des Flusses auf den Nachbarkanten von v mit $f_{e_1} = f(w_1, v), f_{e_2} = f(v, w_2), f_{e_3} = f(w_3, v), f_{e_4} = f(w_4, v)$
5: Berechne ϵ aus Gleichung (52) und schreibe es in die Matrix $\mathcal{E}$
6: **Falls** $\epsilon > \tau$
7: Füge Kanten zu A hinzu
8: **else**
9: Speichere v als gesättigten Knoten

4.1.2 Probleme der erfolgten Implementierung

Das ERG-Verfahren weist insbesondere Probleme in der Laufzeit auf. Die Beschleunigung der Methode von Ford und Fulkerson durch Edmonds und Karp (siehe Abschnitt 2.3.1) kommt dadurch zustande, dass die Länge der augmentierenden Pfade im Laufe der Iterationen des Verfahrens nicht abnimmt, da stets die kürzesten Pfade gewählt werden. Aufgrund der (nicht kantenbasierten) Kapazitätsbedingung wird diese Eigenschaft beim ERG-Verfahren nicht zwangsläufig eingehalten. Dies kann ein Grund für die hohe Laufzeit des Verfahrens und eine nicht zwangsläufig garantierte Konvergenz sein [19].

Zusätzlich ist fraglich, ob die Existenz einer Aufstiegsrichtung im Graphen tatsächlich die Existenz eines s-t-Weges impliziert. Ist dies nicht der Fall, kann dies zu Fehlern in den Ergebnissen führen. Das könnte zum Beispiel in einem zu kleinen maximalen Fluss resultieren, wenn kein weiterer s-t-Weg auffindbar ist, obwohl eine passende Aufstiegsrichtung existiert.

Um den optimalen Wert des maximalen Flusses des Verfahrens zu überprüfen, wurden im Rahmen dieser Masterarbeit schließlich das primale Problem (50) sowie das duale Problem (44)-(47) in CVX implementiert. Dies ist ein Framework für konvexe Programmierung. Der Einsatz von CVX erlaubt es, Zielfunktionen und Nebenbedingungen eines konvexen Optimierungsproblems in der gewohnten MATLAB-Syntax zu programmieren und zu lösen [1]. Experimentell konnte somit erkannt werden, dass das bisherige Verfahren selbst bei kleinen Graphen häufig einen zu kleinen Wert des maximalen Flusses berechnet (siehe beispielsweise Abbildung 17), da es zu früh keine Aufstiegsrichtung mehr finden kann.

4.2 Lösungsansätze

Das Ziel dieser Arbeit stellt eine Untersuchung und Erweiterung des ERG-Verfahrens dar. In den folgenden Abschnitten sollen verschiedene Ansätze zur Lösung der beschriebenen Probleme vorgestellt werden. Alle folgenden Erweiterungen der Implementierung wurden im Rahmen dieser Masterarbeit zur ursprünglichen Programmierung von Cremer [19] hinzugefügt.

4.2.1 Implementierungsdetails

Im Laufe der Analyse der bisher erfolgten Programmierung wurde festgestellt, dass die implementierte Funktion zur Bestimmung der Nachbarknoten nur für quadratische Bilder fehlerfrei umgesetzt wurde. Dies wurde demnach angepasst, so dass nun auch an den äußeren Knoten von nichtquadratischen Bildern die richtigen Nachbarn gefunden werden können.

Zur Vorbeugung vor numerischen Fehlern in der Bestimmung der augmentierenden Pfade in Schritt 6 aus Algorithmus 11 wurde in der bisherigen Implementierung eine feste Toleranzgrenze von $\tau = 10^{-8}$ festgelegt. Somit wurden Knoten als ungesättigt interpretiert, sofern $\epsilon > \tau$ war. Diese Toleranz wurde im Laufe der durch die Masterarbeit erfolgten Tests als Parameter übergeben, um nicht auf eine spezifische Toleranzgrenze festgelegt zu sein.

Schließlich ließ die Implementierung von Cremer [19] in der Rekonstruktion des s-t-Weges in G aus dem s_1-t_1-Pfad des erweiterten Residualgraphen die Quelle und die Senke außer Acht. Demnach wurde der Fluss nicht auf allen Kanten des Pfades erhöht. Einige Kanten mit Verbindungen zur Quelle beziehungsweise Senke wurden nicht in der Augmentierungsphase berücksichtigt. Dies führte dazu, dass der berechnete Fluss häufig zu klein war. Dieser Fehler wurde in dieser Masterarbeit berichtigt.

Zusätzlich darf die Aufstiegsrichtung ϵ nicht negativ sein (siehe Abschnitt 4.1). Die von Cremer [19] erfolgte Implementierung von Algorithmus 11 berücksichtigte diese Tatsache nur für alle Knoten ohne Quelle und Senke. Daher wurde auch dieser Schritt im

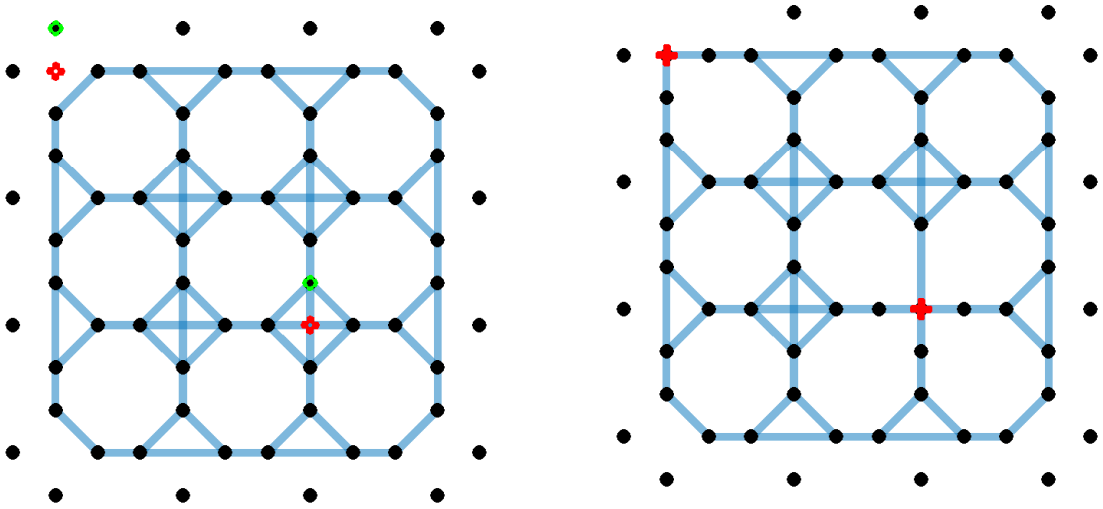
Laufe der Masterarbeit angepasst und die Bedingung auch für die Quelle und die Senke realisiert.

4.2.2 Pfadsuche durch den erweiterten Residualgraphen

Ein weiteres Problem stellte die Suche eines s_1-t_1-Pfades innerhalb des erweiterten Residualgraphen dar. Im bisher implementierten Verfahren wurde der kürzeste Pfad in G_f mithilfe eines eigens programmierten Dijkstra-Verfahrens berechnet. Da die Pfadsuche jedoch innerhalb des Algorithmus' häufig durchgeführt wird und den Großteil der Laufzeit in Anspruch genommen hat, wurde die Pfadsuche durch eine Breitensuche mithilfe der MATLAB-internen Funktion `graphshortestpath` ersetzt, was zu einer Beschleunigung der Laufzeit um den Faktor von vier geführt hat.

Wie bereits in Abschnitt 4.1.1 beschrieben, wurden stets die obersten Unterknoten der Viererblöcke der Quelle und Senke für die Pfadsuche verwendet. Dies kann vor allem problematisch werden, wenn sich die Quelle oder die Senke am Rand des Graphen befindet. Abbildung 11a zeigt als Beispiel einen 4×4-Graphen, bei dem die Quelle als erster und die Senke als elfter Knoten gewählt wurde (dargestellt durch rote Kreuze). Wird für die Suche eines s_1-t_1-Pfades innerhalb des erweiterten Residualgraphen der oberste Unterknoten des Blockes gewählt (grüne Kreise), so wird er in diesem und ähnlichen Beispielen nie erreicht werden, da für die äußeren Knoten von G keine Kanten gleicher Elternknoten berechnet werden. Eine Verbindung zwischen gleichen Elternknoten wird nur dann zum erweiterten Residualgraphen hinzugefügt, wenn in einen Knoten sowohl Fluss ein- als auch ausfließen kann. Der Knoten in der linken oberen Ecke der Abbildung 11a besitzt in dem gewählten Beispiel keinen Fluss, der von oben oder von links einfließen kann. Daher gibt es im erweiterten Residualgraphen keine Verbindung zu den oberen und linken Teilknoten des Viererblocks.

Dieses Beispiel wäre durch die Wahl des oberen Knotens für die Quelle nicht lösbar. Daher wurde der erweitertete Residualgraph im Rahmen der Erweiterungen für diese Masterarbeit für die Quelle und Senke speziell verändert: Es wird weiterhin davon ausgegangen, dass s_1 und t_1 die obersten Unterknoten der Viererblöcke für s und t darstellen. Jedoch werden diese in die Mitte dieser Blöcke verschoben und die Adjazenzmatrix insofern abgewandelt, als dass dieser Knoten die Verbindung aller seiner Nachbarn des Viererblockes übernimmt. Für das oben beschriebene Beispiel ist der abgewandelte erweiterte Residualgraph in Abbildung 11b dargestellt. Quelle und Senke können somit definitiv erreicht werden. Zudem gibt es nun nur eine Möglichkeit für die Wahl der Start- und Endpunkte, was eine eventuelle Fallunterscheidung für bessere Ergebnisse ausschließt.



(a) Erweiterter Residualgraph in der bisherigen Implementierung.

(b) Neuer erweiterter Residualgraph nach korrekter Behandlung von Quelle und Senke.

Abbildung 11. Vergleich des ursprünglichen Aufbaus des erweiterten Residualgraphen (links) mit der neuen Darstellung (rechts). Quelle und Senke des Ursprungsgraphen G sind als rote Kreuze dargestellt. Links sind die heuristischen Wahlen der Quelle und Senke in G_f mit grüne Kreisen verdeutlicht. Werden die obersten der vier Unterknoten in Quelle und Senke als Anfangs- und Endpunkt für die Pfadsuche verwendet, so kann in manchen Fällen fälschlicherweise kein Pfad zu ihnen gefunden werden (siehe Abbildung (a) links oben). Die neue Variante verschiebt den obersten Knoten in die Mitte des Blocks und sorgt dafür, dass in jedem Fall eine Verbindung zu den Nachbarn besteht.

4.2.3 Erweiterung der Pfadsuche nach Boykov und Kolmogorov

Um das ERG-Verfahren weiter zu verbessern, wird in dieser Arbeit die Idee von Boykov und Kolmogorov (siehe [11] oder Abschnitt 2.3.3) aufgegriffen und zwei Suchbäume für die Suche nach Pfaden verwendet. Ziel dabei ist es, die Pfadsuche zu beschleunigen, damit nicht in jedem Schritt ein neuer Pfad berechnet werden muss, sondern Teile des Pfades wiederverwendet werden können.

Abbildung 12 zeigt die Reihenfolge der gesuchten Pfade mithilfe der MATLAB-Funktion `graphshortestpath` und der Erweiterung mit zwei Suchbäumen nach Boykov und Kolmogorov. Die ursprünglich gewählte Pfadsuche berechnet in jedem Schritt einen neuen kürzesten Pfad. Im gewählten Beispiel werden somit zehn Pfade benötigt, um das Optimum des Maximalflussproblems zu erreichen. Obwohl sich viele Knoten in den nacheinander berechneten Pfaden überschneiden, beginnt die Suche in jeder Iteration von Neuem. Die Erweiterung nach Boykov und Kolmogorov kann Teile von Pfaden wiederverwenden.

Obwohl der Algorithmus nur im ersten Schritt das Auffinden des kürzesten Pfades garantieren kann, terminiert er bei dem gleichen Beispiel bereits nach acht Iterationen. Bei einer geeignet effizienten Implementierung wird die Laufzeit in diesem Beispiel somit potentiell reduziert. Die Erweiterung wurde bislang ausschließlich in MATLAB implementiert. Die Verwendung vieler Schleifen macht das Verfahren insbesondere bei großen Testdaten langsam. Bislang wurde keine Umsetzung in C++ durchgeführt, da dies den zeitlichen Rahmen dieser Arbeit sprengen würde.

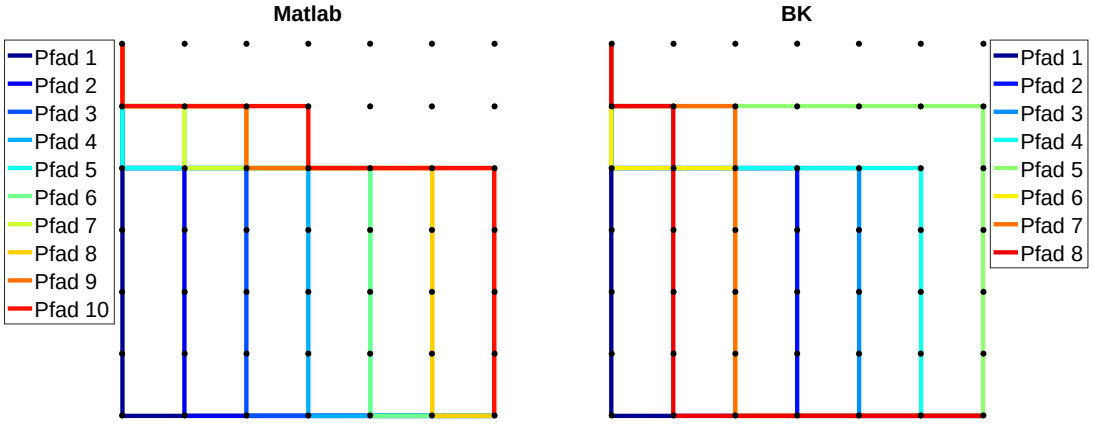


Abbildung 12. Darstellung der Pfadsuche mit verschiedenen Verfahren. Links sind die besuchten Pfade bei einer Breitensuche mithilfe der MATLAB-Funktion `graphshortestpath`, rechts diejenigen bei einer Suche mithilfe zweier Suchbäume nach Boykov und Kolmogorov, dargestellt. Links sind die zehn kürzesten Pfade einer jeden Iteration bestimmt, die benötigt werden, um das Maximum zu erreichen. Rechts werden nicht zwangsläufig die kürzesten Pfade berechnet, dafür werden in diesem Beispiel nur acht Iterationen bis zum Auffinden des Optimums benötigt, was eine potentiell geringere Laufzeit impliziert.

Durch die Implementierung nach dem Verfahren von Boykov und Kolmogorov [11] wurden die Knoten des erweiterten Residualgraphen in aktive, passive und freie Knoten eingeteilt. Die ersten beiden Typen lassen sich eindeutig einem der Suchbäume S und T zuordnen. Dahingegen ist die Einteilung der freien Knoten unklar. In der Implementierung wurde somit zusätzlich eine Unsicherheitsregion auf Basis der freien Knoten bestimmt, die am Ende des Verfahrens noch keinem Suchbaum zugewiesen werden konnten. Dies ist ein Vorteil, da auf dieser Basis weiterführend ein (subpixel-) genaueres Verfahren entwickelt werden kann. In Kapitel 5 wird dies mittels einiger Bilddaten demonstriert.

4.2.4 Kapazitätsbeschränkung für die Quelle und die Senke

Bereits in Abschnitt 4.2.1 wurde erläutert, dass die Quelle und die Senke in der bisher erfolgten Implementierung durch Cremer [19] fehlerhafterweise nicht in die Pfadsuche und die Suche nach einer Aufstiegsrichtung mit einbezogen wurde. Eine genauere Betrachtung des Maximalflussproblems (44)-(47) im Vergleich zu der primalen Energiefunktion (35) und dem dualen Problem (36)-(38) macht deutlich, dass die Kapazitätsbeschränkungen auch in Quelle und Senke gelten müssen. Gleichung (45) erzeugt diese jedoch nicht. Daher ergibt sich das korrekte Maximalflussproblem im Zusammenhang mit dem erweiterten Residualgraphen als

$$\max_{f:E\to\mathbb{R}} \text{div } f_s, \quad \text{so dass} \tag{54}$$

$$f_v \text{ zulässig} \quad \forall v \in V, \tag{55}$$

$$\text{div } f_v = 0, \quad \forall v \in V\backslash\{s,t\} \tag{56}$$

$$Af = 0. \tag{57}$$

Die Kapazitätsbedingung (51) setzt voraus, dass Fluss durch einen Knoten hindurch fließt. Häufig wird angenommen, dass in Quelle und Senke nur in einer Richtung ein Fluss fließt (von der Quelle zur Senke). Daher wird die Schrittweite $\epsilon > 0$ für Quelle und Senke wie folgt berechnet:

$$(f_{e_1} + \epsilon)^2 + f_{e_2}^2 + f_{e_3}^2 + f_{e_4}^2 \leq c_v^2 \tag{58}$$

$$\Leftrightarrow \quad (f_{e_1} + \epsilon)^2 \leq c_v^2 - (f_{e_2}^2 + f_{e_3}^2 + f_{e_4}^2) =: g \tag{59}$$

$$\Leftrightarrow \quad \epsilon = +\sqrt{g} - f_{e_1} \tag{60}$$

Da die Aufstiegsrichtung stets positiv sein muss und die größtmögliche Flusszunahme beschreibt, wird nur die positive Lösung der quadratischen Gleichung (60) betrachtet. Um eine positive Aufstiegsrichtung zu garantieren, wird ϵ für $g \geq 0$ und $\sqrt{g} > f_{e_1}$ als

$$\epsilon = \max(0, \sqrt{g} - f_{e_1}) \tag{61}$$

gewählt. Für eine konsistente Verwendung der Matrix $\mathcal{E}$ aus Abschnitt 4.1.1 werden die Einträge für die Quelle und Senke speziell angepasst. Innerhalb von $\mathcal{E}$ speichert das Element $\mathcal{E}(v, \text{direction}_1, \text{direction}_2)$, um wie viel der Fluss im Knoten v von Richtung direction_1 nach direction_2 weiter erhöht werden kann, ohne die Kapazitätsbeschränkung zu verletzen. Da ein solches Blockelement der Matrix $\mathcal{E}$ für einen Knoten eine 4×4-Matrix darstellt, welche die Schrittweiten für alle Kombinationen der Flussrichtungen durch den Knoten beinhaltet, muss der Block für Quelle und Senke die gleiche Größe haben.

In dieser Masterarbeit wird die Möglichkeit betrachtet, dass Quelle und Senke sowohl Zu- als auch Abfluss besitzen können. Fließt kein Fluss in die Quelle hinein, so wird das entsprechende Matrixelement auf null gesetzt. In diesen spezifischen Knoten gibt es für den Zu- und Abfluss je nur vier Möglichkeiten. Die erste Zeile der Matrizen für

Quelle und Senke beinhaltet daher den Zufluss in allen vier Richtungen, die zweite Zeile den Abfluss. Für die konsistente Darstellung und zur Vermeidung von falschen Aufrufen werden die letzten beiden Zeilen mit `NaN`-Werten gefüllt und man erhält

$$\mathcal{E}(w,\cdot,\cdot) = \begin{pmatrix} \text{Inflow}_1 & \text{Inflow}_2 & \text{Inflow}_3 & \text{Inflow}_4 \\ \text{Outflow}_1 & \text{Outflow}_2 & \text{Outflow}_3 & \text{Outflow}_4 \\ \texttt{NaN} & \texttt{NaN} & \texttt{NaN} & \texttt{NaN} \\ \texttt{NaN} & \texttt{NaN} & \texttt{NaN} & \texttt{NaN} \end{pmatrix} \tag{62}$$

für $w \in \{s,t\}$. Mögliche Schrittweiten in Quelle und Senke können nun aus den ersten beiden Zeilen der jeweiligen Blöcke ausgelesen werden. In den in Kapitel 5 getesteten Fällen besitzt die Quelle nur einen Ab- und die Senke nur einen Zufluss. Dadurch ist die erste beziehungsweise zweite Zeile des Elements $\mathcal{E}(w,\cdot,\cdot)$ in diesen Fällen null.

4.2.5 Quasizyklen

Das Hauptproblem des ERG-Verfahrens liegt darin, dass es zu schnell jene Knoten sättigt, durch die weitere augmentierende Pfade laufen könnten. Dies resultiert in einem kleineren maximalen Wert des Flusses als möglich wäre. Zur Bewertung der Güte des Verfahrens wurde jeder Testdurchlauf mit einer Implementierung des Problems in CVX verglichen (siehe Abschnitt 4.1.2 und 5.1). Bilder bis zu einer Größe von 2×3 lieferten das gleiche Ergebnis. Größere Bilder resultierten im Vergleich zu der Lösung des Optimierungsproblems mit CVX in der Regel in einem kleineren Maximalfluss.

Dies zeigt zudem, dass die Existenz einer Aufstiegsrichtung nicht zwangsläufig einen augmentierenden s-t-Weg innerhalb von G impliziert. Kann eine Aufstiegsrichtung gefunden werden, obwohl bereits alle Knoten insofern gesättigt sind, dass kein Pfad durch den erweiterten Residudalgraphen G_f von der Quelle zur Senke auffindbar ist, ist der resultierende Fluss zu klein. Ausgehend von der vorhandenen Aufstiegsrichtung sollte weiterer Fluss durch den Graphen fließen können. Die zu frühe Sättigung der Knoten, die mögliche s-t-Pfade durch G_f unterbricht, verhindert die Flusszunahme, die durch die berechnete Aufstiegsrichtung noch möglich sein sollte.

Zu diesem Zweck wird im Folgenden ein Ansatz zur Umgehung des Problems vorgestellt, der in dieser Arbeit entwickelt wurde und der anhand von Beispiel 1 hergeleitet wird.

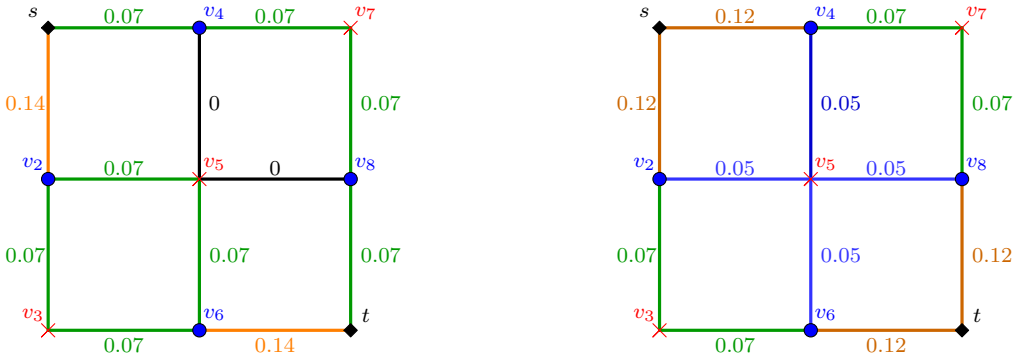
Beispiel 1. Sei $G = (V,E)$ ein Graph mit neun Knoten v_i, $i = 1,\dots,9$ mit drei Zeilen und drei Spalten sowie der Kapazitätsmatrix

$$c = \begin{pmatrix} 10^5 & 0.6 & 0.1 \\ 0.6 & 0.1 & 0.6 \\ 0.1 & 0.6 & 10^5 \end{pmatrix}. \tag{63}$$

Die Quelle wurde als $s = v_1$, die Senke als $t = v_9$ mit $c(s) = 10^5 = c(t)$ gewählt. Das Maximalflussproblem wurde sowohl mithilfe des ERG-Verfahrens als auch durch CVX

gelöst. Den berechneten Fluss beider Methoden zeigt Abbildung 13. Das ERG-Verfahren berechnet den Wert des maximalen Flusses als 0.21, CVX wiederum einen Wert von 0.24. Demnach hat das ERG-Verfahren bereits terminiert, obwohl der optimale Fluss noch nicht gefunden wurde.

Im erweiterten Residualgraph lässt sich kein weiterer Pfad finden, da die Knoten v_3, v_5 und v_7 bereits gesättigt sind, was in der Abbildung durch rote Kreuze dargestellt ist. Daher ist es nun erforderlich zu untersuchen, ob es möglich ist, weiteren Fluss durch den Graphen fließen zu lassen, ohne die Kapazitätsbedingung zu verletzen.



(a) Lösung mithilfe des ERG-Verfahrens.

(b) Lösung mithilfe von CVX.

Abbildung 13. Ergebnis des Maximalflussproblems für den Graphen G aus Beispiel 1 durch Lösung mithilfe des ERG-Verfahrens (links) und CVX (rechts). Die Quelle und die Senke sind durch schwarze Rauten gekennzeichnet, gesättigte Knoten durch rote Kreuze. Beide Verfahren berechnen einen unterschiedlichen Fluss, wobei die Lösung von CVX einen größeren – und optimalen – Wert findet als das ERG-Verfahren.

Zunächst wird die Differenz des Flusses der Lösung mittels CVX und derjenigen durch das ERG-Verfahren betrachtet. Das Resultat ist in Abbildung 14 dargestellt. Auffällig sind dabei zwei entstehende Pfade, welche in der Abbildung in grün und blau dargestellt sind. Der in grün abgebildete Pfad zeigt eine mögliche Flusszunahme um 0.05. Indes repräsentiert der blaue Pfad eine Abnahme des Flusses um 0.02. Beide Pfade durchlaufen den bereits gesättigten Knoten v_5. Der grüne Pfad würde zu einer Übersättigung dieses Knotens führen, die jedoch durch die zusätzliche Abnahme des Flusses auf dem blauen Rückpfad verhindert wird. Der Knoten v_5 verfügt über eine Kapazität von $c(5) = 0.1$. Wird nun über die Kanten f_5^{1-} und f_5^{2+} ein Fluss von 0.05 hinzugefügt und über f_5^{2-} und f_5^{1+} der Wert 0.02 abgezogen, ergibt dies für die Kapazitätsbedingung (51)

$$\begin{aligned}&(f_5^{2-})^2 + (f_5^{1+})^2 + (f_5^{2+})^2 + (f_5^{1-})^2 \\ &= (0 + 0.05)^2 + (0 + 0.05)^2 + (0.07 - 0.02)^2 + (0.07 - 0.02)^2 \\ &= 0.01 = 0.1^2 = c(5)^2.\end{aligned} \tag{64}$$

Durch die gleichzeitige Zunahme und eine etwas kleinere Abnahme des Flusses in Knoten v_5 wird die Kapazität weiterhin nicht überschritten. Diese Verschiebung konnte allerdings eine Zunahme des Flusses von der Quelle zur Senke um $0.05 - 0.02 = 0.03$ erreichen: Danach entspricht der Wert des Flusses mit $0.21 + 0.03 = 0.24$ dem Maximalfluss, der von CVX berechnet wurde.

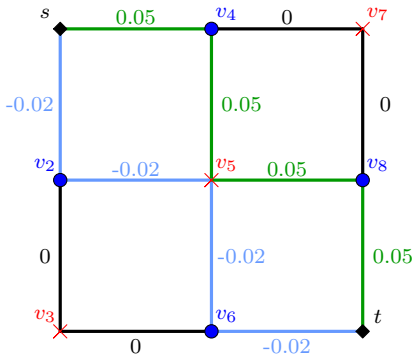


Abbildung 14. Differenz der Ergebnisflüsse durch die Lösung mithilfe von CVX und dem ERG-Verfahren. In grün und blau sind zwei Pfade von der Quelle zur Senke mit den gleichen Differenzwerten dargestellt.

Definition 14. Sei $N = (G, c, s, t)$ mit $G = (V, E)$ ein Netzwerk und G_f der zu G gehörige erweiterte Residualgraph. Dann bezeichne P_{forward} einen s-t-Pfad über den gesättigten Knoten v_x mit einer Flusszunahme von $\epsilon_{\text{forward}} \geq 0$. Zusätzlich sei P_{backward} ein t-s-Pfad über den gesättigten Knoten v_x mit einer Flussabnahme von $\epsilon_{\text{backward}}$, wobei $|\epsilon_{\text{backward}}| < |\epsilon_{\text{forward}}|$.
Ergibt die Flusszunahme von $\epsilon_{\text{forward}}$ über P_{forward} und die Flussabnahme von $\epsilon_{\text{backward}}$ über P_{backward} eine Erhöhung des Maximalflusses in G bei Einhaltung der Kapazitätsbedingung, so werden P_{forward} und P_{backward} als *Quasizyklus* bezeichnet.

Kann ein solcher Quasizyklus gefunden werden, obwohl im erweiterten Residualgraphen kein s-t-Pfad mehr auffindbar ist, so ist es möglich, den Maximalfluss weiter zu erhöhen. Die Einschränkung $|\epsilon_{\text{backward}}| < |\epsilon_{\text{forward}}|$ aus Definition 14 stellt sicher, dass im Gesamten eine Erhöhung des Flusses erfolgt. Eine größere Ab- als Zunahme des Flusses würde dessen Wert noch weiter verkleinern und somit dem Ziel des Findens eines Quasizyklus' widersprechen. Entspricht der Absolutwert von $\epsilon_{\text{backward}}$ dem von $\epsilon_{\text{forward}}$, so spricht dies für eine ungünstige Wahl von P_{forward}, die keine Auswirkung auf den Maximalfluss hat, da der Fluss den gleichen Wert zu- und abnimmt.

Suche von Quasizyklen
Die Verschiebung des Flusses durch eine Flusszunahme und -abnahme innerhalb eines Quasizyklus' erlaubt potentiell eine bessere Approximation des maximalen Flusses durch die zusätzlich mögliche Aufstiegsrichtung. Dazu müssen jedoch Quasizyklen gefunden werden, was im Falle größerer Testdaten nicht trivial ist.

Annahme 1. Seien ein geeigneter s-t-Pfad P_{forward} sowie eine Flusszunahme $\epsilon_{\text{forward}}$ bekannt. Dann können ein passender t-s-Pfad P_{backward} und eine Flussabnahme $\epsilon_{\text{backward}}$ gefunden werden, die zusammen mit P_{forward} einen Quasizyklus bilden.

Für die Suche nach dem Rückpfad P_{backward} wird zunächst der Fluss entlang P_{forward} um $\epsilon_{\text{forward}}$ erhöht. Somit ist ein Knoten v_x, der bereits die Kapazitätsgrenze erreicht hatte, übersättigt. Gesucht ist eine Schrittweite $\epsilon_{\text{backward}}$, die den Fluss durch den Knoten wieder auf die Kapazität von v_x reduziert. Um eine geeignetes $\epsilon_{\text{backward}}$ zu finden, wird die Kapazitätsbedingung (52) gelöst und eine Schrittweite gesucht, welche die Übersättigung aufhebt. Ausgehend von der berechneten Schrittweise kann ein Pfad P_{backward} konstruiert werden. Die Vorgehensweise ist analog zu der in Abschnitt 4.1.1 erläuterten, daher wird an dieser Stelle auf eine detaillierte Beschreibung verzichtet.

Es ist folglich möglich, einen passenden Rückpfad sowie eine Abnahme des Flusses zu berechnen, sofern ein geeigneter s-t-Pfad vorhanden ist, über den zusätzlicher Fluss geleitet werden kann. Es bleibt jedoch die Frage, wie ein solcher Hinpfad automatisiert berechnet werden kann.

Kapitel 5: Ergebnisse

In diesem Kapitel werden die Ergebnisse des vorgestellten ERG-Verfahrens für verschiedene Kriterien präsentiert.

- Der erste Abschnitt bezieht sich auf die Optimalität des ERG-Verfahrens im Vergleich zur exakten Lösung mit CVX.
- Anschließend wird anhand einiger Beispiele die Isotropie der kontinuierlichen Methode der Anisotropie der diskreten Verfahren gegenüber gestellt.
- Schließlich folgt eine Analyse der Laufzeit, um die kontinuierlichen Verfahren mit den diskreten zu vergleichen.

Die Prüfung der ersten beiden Kriterien erfolgt sowohl für synthetische als auch für reale Bilddaten. Die Laufzeitanalyse wird für ein reales Bild in verschiedenen Größen durchgeführt, um die Skalierung der Laufzeit in Abhängigkeit der Bildgröße zu betrachten.

5.1 Optimalität

Im ersten Schritt erfolgt eine Beurteilung der Optimalität des Verfahrens. Wie in Kapitel 4 angedeutet, wird das globale Optimum des Maximalflussproblems in einigen Fällen nicht erreicht. Im Folgenden sollen anhand von synthetischen und realen Bildern zunächst zwei Beispiele gezeigt werden, für die das globale Optimum berechnet werden kann. Anschließend werden Bilder vorgestellt, bei denen der berechnete Maximalfluss zu klein ist.

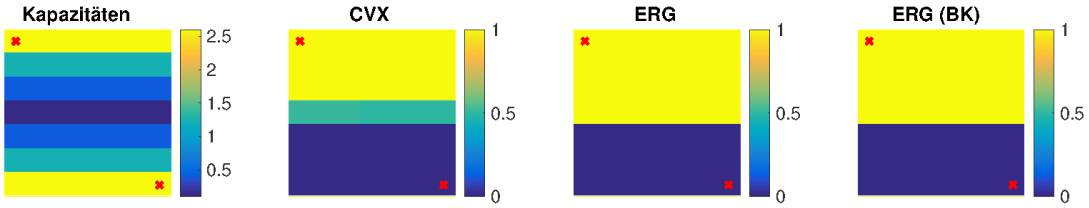


Abbildung 15. Berechnung eines minimalen Schnittes. Die Quelle und Senke sind durch rote Kreuze dargestellt. Von links: Künstlich gewählte Kapazitäten mit hohen Kosten in gelb, niedrigen in blau. Darauf folgt die Lösung des Problems mittels CVX, dem ERG-Verfahren mit gewöhnlicher Pfadsuche und einer Pfadsuche nach Boykov und Kolmogorov. CVX bestimmt zusätzlich eine unsichere Region der Zuordnung. Alle Verfahren berechnen denselben optimalen Maximalfluss.

Für die künstlich erstellten Bilder wurden die Kosten $c \in \mathbb{R}$ mittels

$$c(x, y) = |x \cos(\alpha) + y \sin(\alpha)|^k, \alpha \in [0, 2\pi), k \in \mathbb{N} \tag{65}$$

berechnet. In Abbildung 15 wurden $\alpha = 0$ und $k = 2$ gewählt. In diesem Fall entspricht der optimale Schnitt einer Horizontalen. Hier wird auch ohne die Erweiterung durch

einen Quasizyklus aus Abschnitt 4.2.5 ein maximaler Fluss berechnet. Abbildung 15 zeigt den Vergleich der Segmentierungen mithilfe von CVX und dem ERG-Verfahren mit unterschiedlichen Methoden zur Suche der benötigten Pfade. Die Pfadsuche mithilfe der MATLAB-internen Funktion `graphshortestpath` wird im Folgenden weiterhin als ERG bezeichnet, eine Suche mittels zweier Suchbäume nach Boykov und Kolmogorov [11] als ERG (BK).

Die auf den Bildpunkten gewählten Kosten sind links in der Abbildung dargestellt und in blau für niedrige und in gelb für hohe Werte visualisiert. Die Berechnung des minimalen Schnittes in CVX liefert im Gegensatz zum hier beschriebenen Verfahren einen Übergangsbereich (hellblau), in dem die Zuordnung der Bildpunkte zu einem Segment nicht eindeutig ist. Alle drei Verfahren berechnen einen maximalen Fluss von 0.49 und somit das globale Optimum des Maximalflussproblems.

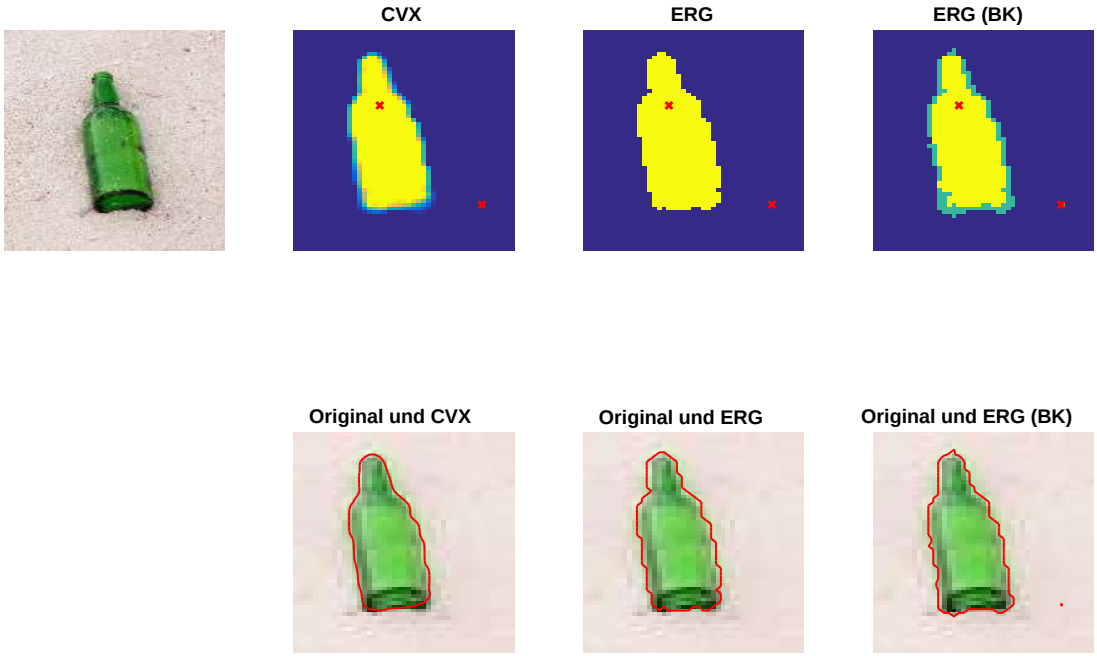


Abbildung 16. Segmentierung des Originalbildes links oben. Um eine Berechnung mit allen Verfahren zu ermöglichen, wurde die Größe des Bildes reduziert. Die Quelle und Senke sind durch rote Kreuze dargestellt. In der ersten Zeile sind die Ergebnisse von CVX, des ERG-Verfahrens mit gewöhnlicher und mit erweiterter Pfadsuche zu sehen. Die untere Reihe zeigt die Konturen (rot) der Segmentierungsergebnisse im Originalbild. Sowohl CVX als auch die Erweiterung des ERG-Verfahrens durch Suchbäume können eine unsichere Region (hellblau) darstellen. Alle Verfahren berechnen denselben optimalen Maximalfluss. Quelle des Bildes: Pixabay.

Um die Optimalität des ERG-Verfahrens auch für reale Bilddaten zu testen, wurde die Bildgröße auf 60×45 Pixel verringert, so dass eine Berechnung auch mittels CVX möglich wurde. Aufgrund von Speicherproblemen kann diese Methode keine größeren Probleme lösen. Die Kapazitäten für reale Bilddaten wurden durch

$$c = \frac{1}{1 + \alpha \cdot \exp(\beta \cdot |\nabla I|)} \tag{66}$$

berechnet, wobei I das Eingabebild bezeichnet. Angelehnt an [19] wurden die Parameter als $\alpha = 100$ und $\beta = 110$ gewählt.
Für das Bild aus Abbildung 16 wurde von allen drei Verfahren ein Maximalfluss von 0.002 berechnet. Die Erweiterung des ERG-Verfahrens nach Boykov und Kolmogorov lässt zusätzlich zu dem Segmentierungsergebnis eine Unsicherheitsregion bestimmen. Diese wurde hier ähnlich zu der von CVX berechneten zum Ergebnisbild hinzugefügt (siehe Abschnitt 4.2.3). Die untere Zeile der Abbildung 16 stellt zudem die Konturen der Segmentierungen durch die beiden ERG-Varianten im verkleinerten Originalbild dar. Die Erweiterung mit zwei Suchbäumen zeigt ohne den Unsicherheitsbereich das gleiche Ergebnis wie das gewöhnliche ERG-Verfahren. Alle Verfahren berechnen gute Segmentierungen.

Wird die Schnittgerade der künstlich generierten Bilder aus Abbildung 15 um einen Winkel von $\frac{3\pi}{5}$ gedreht, findet das ERG-Verfahren nicht die optimale Lösung. Der von CVX berechnete Maximalfluss besitzt einen Wert von 0.78, das ERG-Verfahren jedoch nur 0.71. In der Abbildung 17 lässt sich erkennen, dass dessen Schnitt jedoch noch innerhalb der von CVX berechneten Unsicherheitsregion liegt.

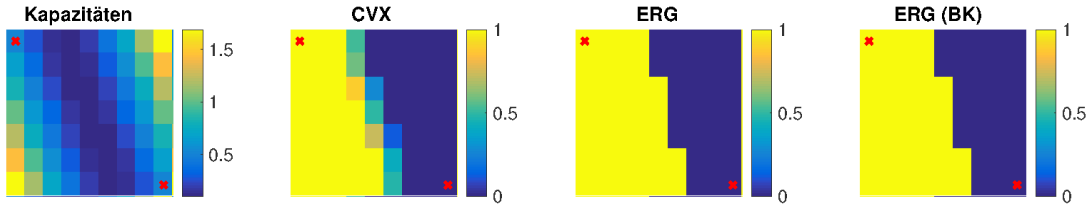


Abbildung 17. Berechnung eines minimalen Schnittes. Die Quelle und Senke sind durch rote Kreuze dargestellt. Von links: Künstlich gewählte Kapazitäten mit hohen Kosten in gelb, niedrige Kosten in blau. Lösung des Problems mittels CVX, dem ERG-Verfahren mit gewöhnlicher Pfadsuche und einer Pfadsuche nach Boykov und Kolmogorov. CVX bestimmt zusätzlich eine unsichere Region der Zuordnung. Beide ERG-Verfahren weisen dieselbe Lösung auf, jedoch ist der berechnete Fluss kleiner als das von CVX berechnete optimale Ergebnis.

Auch bei komplexeren realen Bilddaten weist das ERG-Verfahren häufig einen zu kleinen Maximalfluss auf. Ein Beispiel zeigt Abbildung 18. Die Kapazitäten wurden wie in Gleichung (66) gewählt. Abbildung 18 stellt die Segmentierungsergebnisse der beiden ERG-Varianten und CVX (obere Zeile) dar. Der mithilfe der Erweiterung der Pfadsuche

dargestellte Unsicherheitsbereich des ERG-Verfahrens zeigt zusätzlich zu der Segmentierung der Blüte die Umrisse des im Bild zu sehenden Kaktus', welcher auch in der CVX-Segmentierung zu erahnen ist. Zusätzlich gibt es auch einige Unsicherheiten innerhalb des Kaktus'. Diese kommen vermutlich aufgrund der helleren Stachel zustande, dessen Intensitätswerte denen der segmentierten Blüte ähneln. Das Bild wurde auf eine Größe von 80×53 Pixeln reduziert, um eine Berechnung mit allen Verfahren zu ermöglichen.

Auch wenn die Segmentierungsergebnisse sehr ähnlich zu der optimalen Lösung sind, weisen sie jedoch im Vergleich zu CVX einen zu kleinen Maximalfluss auf. CVX berechnet einen Wert von 0.017, das Ergebnis der ERG-Varianten jedoch nur 0.008.

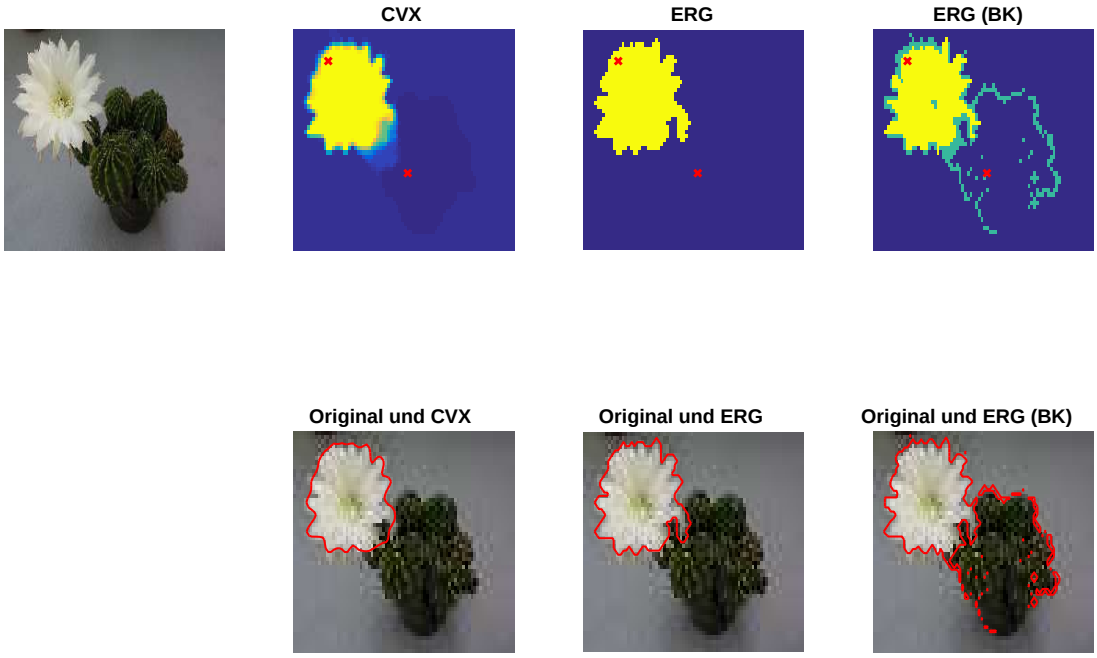


Abbildung 18. Segmentierung des Originalbildes links oben. Die Quelle und Senke sind durch rote Kreuze dargestellt. Um eine Berechnung mit allen Verfahren zu ermöglichen, wurde die Größe des Bildes reduziert. In der ersten Zeile sind die Ergebnisse von CVX, des ERG-Verfahrens mit gewöhnlicher und erweiterter Pfadsuche zu sehen. Die untere Reihe zeigt die Konturen (rot) der Segmentierungsergebnisse im Originalbild. Sowohl CVX als auch die Erweiterung des ERG-Verfahrens durch Suchbäume sind in der Lage, eine unsichere Region darzustellen. Beide ERG-Verfahren berechnen einen zu kleinen Maximalfluss im Vergleich mit dem optimalen Wert durch CVX. Quelle des Bildes: Wikimedia Commons.

5.2 Isotropie

Bereits in Abschnitt 2.8 wurde das Problem der Anisotropie bei diskreten Graph-Cut-Verfahren beschrieben. Dadurch werden bestimmte Richtungen des minimalen Schnitts bevorzugt, die nicht optimal sind. Dieser Abschnitt zeigt den Vergleich der Ergebnisse verschiedener Ansätze. Dazu werden folgende Verfahren betrachtet:

1. CVX
2. ERG-Verfahren
3. ERG-Verfahren mit Erweiterung nach Boykov und Kolmogorov [11]
4. Graph-Cut-Verfahren mit einer 4er-Nachbarschaft
5. Graph-Cut-Verfahren mit einer 16er-Nachbarschaft

Die diskreten Graph-Cut-Verfahren wurden ebenfalls nach Boykov und Kolmogorov [11] implementiert. Abbildung 19 zeigt den Unterschied des 4er- und des 16er-Nachbarschaftssystems. Der Übergang zu einer größeren Nachbarschaft kann verwendet werden, um Diskretisierungsartefakte verringern [10]. Allerdings vergrößert sich dadurch der Graph und es können weniger feine Details aufgelöst werden. Die Vergrößerung der Nachbarschaft kann zudem zu Ungenauigkeiten führen, wenn Bildpunkte miteinander verbunden werden, die so weit voneinander entfernt sind, dass ihre inhaltliche Zusammengehörigkeit fraglich ist.

(a) Graph mit 4er-Nachbarschaft. **(b)** Graph mit 16er-Nachbarschaft.

Abbildung 19. Graph mit verschiedenen Nachbarschaftssystemen. Eine größere Nachbarschaft kann Diskretisierungsartefakte verringern, jedoch auf Kosten der Genauigkeit der Segmentierung.

Die Anisotropie der diskreten Verfahren ist beispielsweise in Abbildung 20 zu erkennen. Diese zeigt den optimalen Schnitt als Gerade in verschiedenen Orientierungen. Von links nach rechts sind die Ergebnisse der fünf Verfahren dargestellt. Beide ERG-Verfahren weisen bis auf einige Unregelmäßigkeiten einen zu CVX ähnlichen glatten Verlauf auf.

Abbildung 20. Segmentierung mit verschiedenen Lösungsverfahren für Geraden in verschiedenen Rotationen. Die Quelle und Senke sind durch rote Kreuze dargestellt. Die Kosten sind in der ersten Spalte für die Winkel $\alpha \in \{0.1\pi, 0.2\pi, 0.3\pi, 0.4\pi\}$ (von oben nach unten) und $k = 3$ der Gleichung (65) dargestellt. Hohe Kosten sind in gelb, niedrige in blau angezeigt. Von links: CVX, ERG-Verfahren mit Pfadsuche mittels MATLAB, ERG-Verfahren erweiterter mit Pfadsuche, diskrete Graph-Cut-Verfahren mit einer 4er-Nachbarschaft und einer 16er-Nachbarschaft. CVX und die Erweiterung des ERG-Verfahrens weisen zusätzlich eine unsichere Region auf. In schwarz sind die berechneten Segmentgrenzen der verschiedenen Ansätze zu sehen. Die äußeren Grenzen des Unsicherheitsbereichs von CVX sind über den Ergebnissen der übrigen Verfahren in rot dargestellt. Das ERG-Verfahren zeigt recht ähnliche Ergebnisse, jedoch sind die Ränder für $\alpha \in \{0.1\pi, 0.2\pi\}$ etwas uneben, da einige Pixel zu viel segmentiert wurden. Das ERG-Verfahren mit der Erweiterung nach Boykov und Kolmogorov berechnet ähnliche Ergebnisse mit einigen Unregelmäßigkeiten im berechneten Unsicherheitsbereich (hellgrün). Das diskrete Verfahren mit einer 4er-Nachbarschaft weist eine deutliche Präferenz zu achsenparallelen Kanten. Eine Vergrößerung des Graphen auf 16 Nachbarn kann diese Artefakte lindern, jedoch können weniger Details aufgelöst werden.

Insbesondere innerhalb der zusätzlich berechneten Unsicherheitsregion (hellgrün) der Erweiterung des ERG-Verfahrens sind einige Ausreißer zu erkennen. Diese kommen vermutlich aufgrund ähnlich hoher Kosten in diesen Bereichen zustande.
Die Abbildung zeigt zusätzlich zu den berechneten Segmentierungen die von CVX bestimmen Segmentgrenzen und dessen unsicheren Bereich. Das gewöhnliche ERG-Verfahren berechnet den Schnitt für $\alpha = 0.1\pi$ im rechten Teil des Bildes leicht oberhalb von CVX, zeigt aber in den anderen Fällen eine gute Annäherung der optimalen Trennung. Die Erweiterung des ERG-Verfahrens stellt ebenfalls eine gute Approximation dar, jedoch sind die Konturen des Unsicherheitsbereich deutlich unebener als bei der gewöhnlichen Variante.

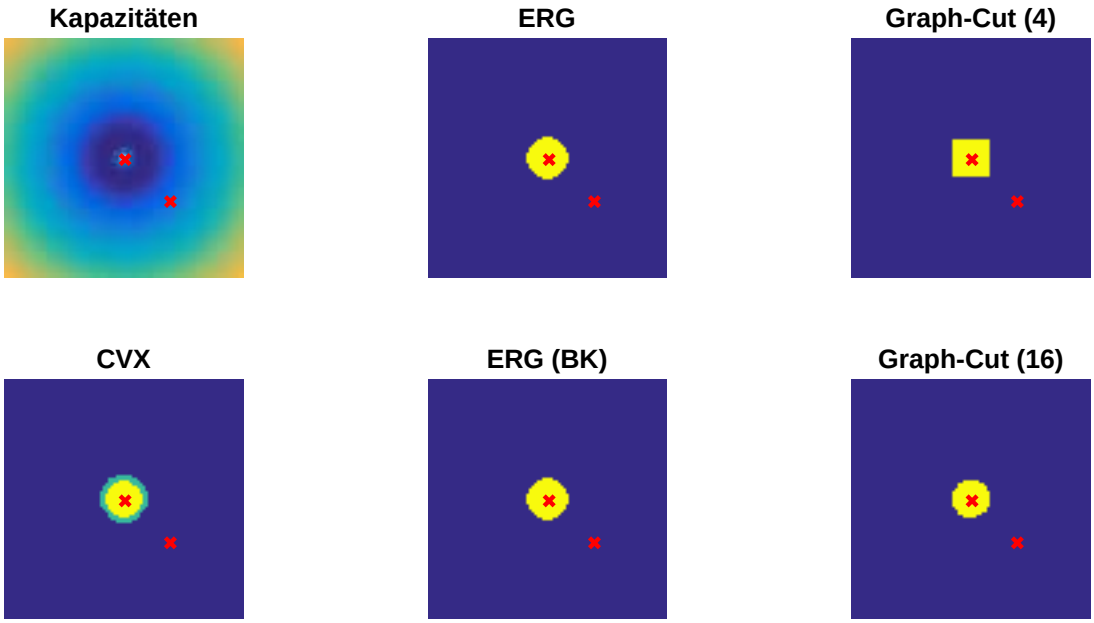


Abbildung 21. Segmentierung eines Kreises. Die Quelle und Senke sind durch rote Kreuze dargestellt. Links oben sind die gewählten Kosten zu sehen. Hohe Kosten sind in gelb, niedrige in blau kodiert. Darunter ist das optimale Ergebnis durch CVX dargestellt. Die zweite Spalte zeigt die Segmentierungen durch beide ERG-Varianten, die letzte Spalte die Ergebnisse der diskreten Ansätze mit einer 4er- und 16er-Nachbarschaft. Die Anisotropie des diskreten Verfahrens äußert sich insofern, dass mit einer 4er-Nachbarschaft ein Quadrat statt des gewünschten Kreises berechnet wird. Der Übergang zu einer größeren Nachbarschaft weist eine kreisförmige Segmentierung auf.

Das diskrete Verfahren nach Boykov und Kolmogorov mit einer 4er-Nachbarschaft zeigt eine deutliche Präferenz für achsenparallele Kanten. Für die Winkel $\alpha = 0.2\pi$ und $\alpha = 0.3\pi$ ist die Andeutung einer schrägen Trennung der Quelle und Senke zu sehen, wobei die Randbereiche weiterhin achsenparallel sind. Für $\alpha = 0.1\pi$ und $\alpha = 0.4\pi$ entspricht der Schnitt einer horizontalen beziehungsweise vertikalen Gerade. Eine Verbesserung konnte durch die Erweiterung auf die größere Nachbarschaft erreicht werden.

Wird statt Linien wie in [40] ein Kreis segmentiert, so zeigt Abbildung 21 einen großen Unterschied zwischen kontinuierlichen und diskreten Verfahren. Während sowohl CVX als auch die beiden ERG-Varianten den gewünschten Kreis ausschneiden, berechnet das diskrete Verfahren mit einer 4er-Nachbarschaft lediglich ein Quadrat. An dieser Stelle wird der Unterschied der isotropen kontinuierlichen und der anisotropen diskreten Verfahren besonders deutlich. Der Übergang zu einer größeren Nachbarschaft mit 16 Nachbarn in jedem Knoten berechnet den gewünschten Kreis.

Für größere oder reale Bilddaten ist ein solcher Unterschied häufig weniger stark erkennbar. Abbildung 22 zeigt die Segmentierung des Kameramanns durch die verschiedenen Verfahren. Die von CVX berechnete Lösung weist nur im Bereich des Körpers eine eindeutige Zuordnung auf. Der Kopf, das Kamerastativ und der Hintergrund sind in einen Unsicherheitsbereich mit verschiedenen „Wahrscheinlichkeiten“ aufgeteilt. Je höher die Wahrscheinlichkeit einer Zuordnung zum Segment des Körpers ist, desto heller wird ein Bereich dargestellt. Die Kontur des Kameramanns ist dennoch erkennbar. Das ERG-Verfahren liefert eine eindeutigere Partition. Jedoch wird hier das Gesicht nicht der gleichen Menge wie dem Rest des Körpers zugeteilt. Dagegen ist die Einteilung dieses Bildbereichs durch die Erweiterung des ERG-Verfahrens nicht eindeutig. Auch das Kamerastativ und Teile des Hintergrunds liegen aufgrund von ähnlichen Intensitätswerten in einer unsicheren Region.

Auch wenn die kontinuierlichen Ergebnisse in diesem Fall einige Unterschiede aufweisen, ist keine erkennbare Richtungsabhängigkeit zu finden. Das diskrete Verfahren mit einer 4er-Nachbarschaft zeigt jedoch eine Präferenz einer achsenparallelen Kontur im rechten Bereich der Segmentierung. Wird die Nachbarschaft vergrößert, so ist die Annäherung an die gewünschte Kontur zwar besser, jedoch verliert das Ergebnis auch an Genauigkeit: Das Objektiv der Kamera wird nicht in die Segmentierung mit einbezogen. Es ist zudem auffällig, dass die Kontur des Arms im linken Bildbereich bei beiden diskreten Methoden zu weit unten liegt.

Abbildung 22. Berechnung einer Segmentierung des Originalbildes links oben. Die Quelle und Senke sind durch rote Kreuze dargestellt. Das optimale Ergebnis durch CVX ist oben in der Mitte dargestellt. In der zweiten Zeile sind die Ergebnisse des ERG-Verfahrens mit gewöhnlicher und erweiterter Pfadsuche sowie mithilfe des diskreten Verfahrens von Boykov und Kolmogorov mit einer 4er- und einer 16er-Nachbarschaft zu sehen. Die Konturen der Segmentierungen der ERG- und der diskreten Verfahren sind über dem Originalbild in der letzten Zeile dargestellt. CVX und die Erweiterung des ERG-Verfahrens berechnen zusätzlich eine Unsicherheitsregion. Keins der kontinuierlichen Verfahren weist eine erkennbare Richtungsabhängigkeit auf. Diese ist allerdings in der Lösung der diskreten Methode mit einer 4er-Nachbarschaft im rechten Bereich der Kontur zu sehen. Die Vergrößerung der Nachbarschaft kann diese Artefakte umgehen, schließt jedoch das äußere Ende der Kamera in der Segmentierung aus.

5.3 Laufzeit

Schließlich erfolgt ein Vergleich der verschiedenen Verfahren in Hinblick auf die Laufzeit. Betrachtet werden wiederum die Ergebnisse von CVX, beiden ERG-Varianten und der diskreten Implementierung mit einer 4er- und einer 16er-Nachbarschaft. Dazu wurde eine Segmentierung des Bildes aus Abbildung 23 mit den beschriebenen Algorithmen durchgeführt. Die Segmentierungsergebnisse aller Methoden sind recht genau und weisen in kleinen Details Unterschiede auf. Um auch einen Vergleich mit CVX zu ermöglichen, wurde das Bild auf eine Größe von 72×96 Pixeln skaliert.

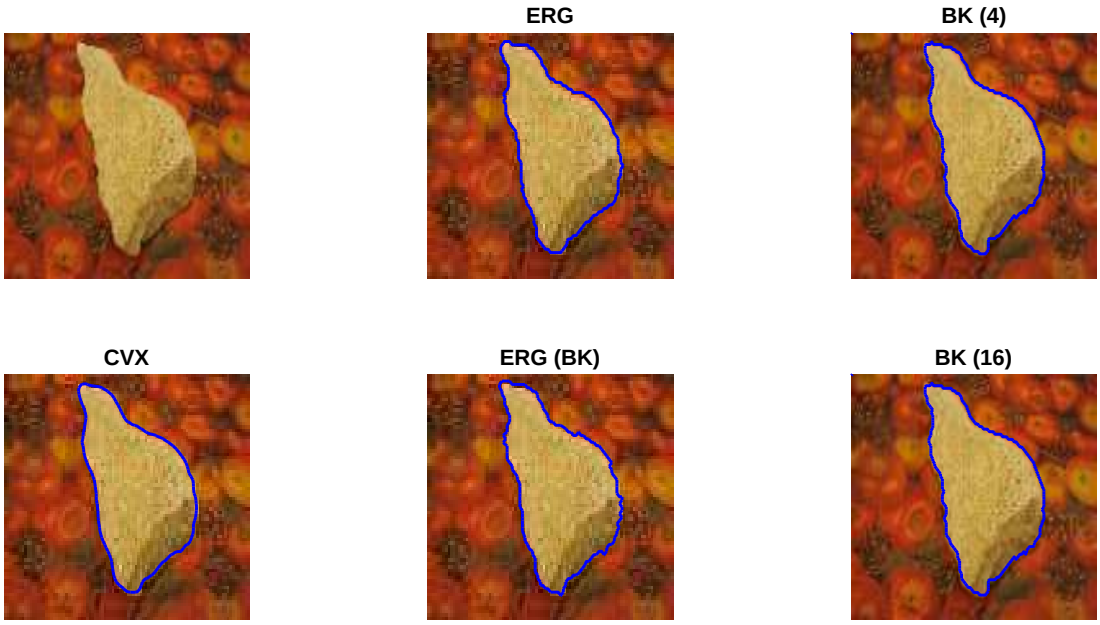


Abbildung 23. Berechnung einer Segmentierung mit verschiedenen Verfahren. Links oben ist das Originalbild dargestellt. Darunter ist das optimale Ergebnis durch CVX zu sehen. Die zweite Spalte zeigt die Segmentierungen durch beide ERG-Varianten, die letzte Spalte die diskreten Ergebnisse mit einer 4er- und 16er-Nachbarschaft. Alle Segmentgrenzen wurden blau im Originalbild dargestellt und zeigen gute Ergebnisse. Quelle des Bildes: `http://vision.middlebury.edu/MRF/`.

Die schnellste Berechnung in 5 Sekunden erfolgte durch das diskrete Verfahren von Boykov und Kolmogorov. Es wurden jedoch mit 700 Pfaden viele Iterationen benötigt, um eine Lösung zu berechnen. Mithilfe von CVX wurde das Problem in 9 Sekunden gelöst. Als Lösungsmethode wurde ein Interior-Point-Verfahren mit 29 Iterationen verwendet. Nur wenig langsamer ist mit 10 Sekunden die diskrete Methode mit einer 16er-Nachbarschaft. Die Vergrößerung des Graphen wirkt sich allerdings negativ auf die Anzahl der Iterationen aus. Die vielen neuen Möglichkeiten einen Pfad zu bestimmen resultieren mit 982 Iterationen in wesentlich mehr Schritten als von den anderen Verfahren benötigt wurden.

Durch die Implementierung des Großteils des ERG-Verfahrens in MATLAB ist das Verfahren gegenüber der C++-Implementierung des diskreten Verfahrens langsamer. Die MATLAB-interne Pfadsuche weist mit 16 Sekunden eine mehr als dreimal so hohe Berechnungszeit auf wie die des diskreten Verfahrens mit einer 4er- Nachbarschaft. Allerdings werden nur 179 Pfade berechnet. Durch die Erweiterung auf eine Suche nach Pfaden mit Suchbäumen werden zwar kontextsensitive Ergebnisse erzielt, jedoch wurde das Verfahren bislang ausschließlich in MATLAB programmiert, was zu einer äußerst langen Rechenzeit von 130 Sekunden führt. Die veränderte Pfadsuche variiert ebenfalls die Reihenfolge der Pfade, was zu mehr Schritten führen kann. In diesem Beispiel wurden 213 Pfade berechnet.

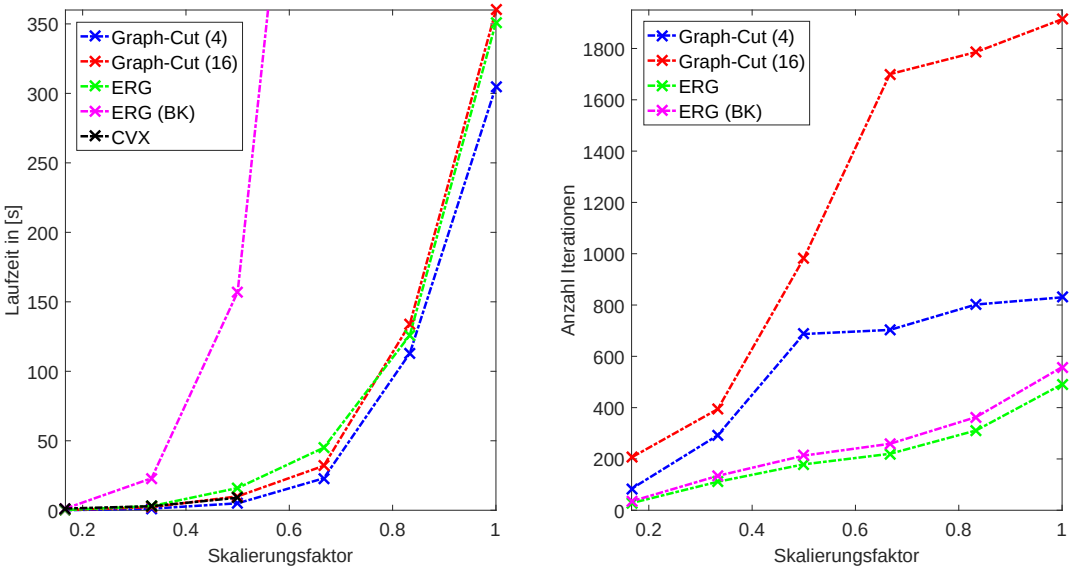


Abbildung 24. Darstellung der Laufzeiten (links) und Anzahl der Iterationen (rechts) der verschiedenen Verfahren für unterschiedliche Größen des Bildes. Das diskrete Verfahren (blau) weist die kürzeste Laufzeit auf mit wenig Abstand zu CVX (schwarz), welches geringfügig langsamer ist. Allerdings kann CVX keine Lösung für größere Bilddaten berechnen und wird daher nur bis zu einer Skalierung von 0.5 aufgetragen. Die Vergrößerung des Graphen des diskreten Verfahrens auf 16 Nachbarn liegt bei kleinen Bilddaten dicht an der Laufzeit von CVX. Anschließend folgt das ERG-Verfahren (grün) und mit einer wesentlich höheren Laufzeit als die anderen Methoden die Erweiterung mit zwei Suchbäumen (rosa). Ab einer Bildgröße von $\frac{2}{3}$ der Orignialgröße ist das ERG-Verfahren schneller als die diskrete Methode mit einer 16er-Nachbarschaft. Im Gegensatz dazu benötigen die kontinuierlichen Verfahren wesentlich weniger Iterationen bis zu einem optimalen Ergebnis als die diskreten Methoden. Vor allem mit einer 16er-Nachbarschaft steigt die Anzahl der benötigten Schritte mit wachsender Bildgröße stark an.

Um die Auswirkungen der Größe des Bildes auf die Laufzeit zu untersuchen, wurde die Segmentierung des Bildes aus Abbildung 23 für verschiedene Skalierungen der Größe durchgeführt. Für CVX wurde dies nur bis zu einem Skalierungsfaktor von 0.5 berechnet, da es für größere Daten aufgrund von Speicherproblemen nicht geeignet ist. Das Originalbild mit einer Größe von 144×192 Pixeln wurde in Schritten von $\frac{1}{6}$ auf eine Größe von 24×32 Pixeln skaliert. Die Laufzeiten sind in Abbildung 24 gegenübergestellt. Das diskrete Verfahren ist wie zu erwarten am schnellsten. Jedoch ist die Laufzeit des ERG-Verfahrens insbesondere für große Bilddaten nur geringfügig langsamer. Bis zu einer Skalierung von 50 % der Originalgröße liegt die Laufzeit von CVX zwischen denen der beiden Verfahren. Eine Vergrößerung des Graphen bei den diskreten Verfahren verlangsamt die Laufzeit. Bis zu einer Skalierung von $\frac{2}{3}$ der Originalgröße liegt die Laufzeit dieser Methode zwischen derjenigen der 4er-Nachbarschaft im diskreten und dem ERG-Verfahren. Bei größeren Bildern ist dieses kontinuierliche Verfahren sogar schneller. Die Erweiterung des ERG-Verfahrens weist eine deutlich höhere Laufzeit auf und ist für größere Bilder kaum praktikabel. Für die Originalgröße benötigte es 76 Minuten für die Segmentierung, wobei das ERG-Verfahren nur eine Laufzeit von 6 und die diskrete Methode von 5 Minuten aufweist.

Dahingegen benötigen alle kontinuierlichen Verfahren weniger Schritte zum Erreichen des Optimums als die diskreten Methoden. Bei maximaler Bildgröße weist das diskrete Verfahren mit einer 16er-Nachbarschaft eine mehr als sechsfache Anzahl an Iterationen auf. Beide ERG-Verfahren besitzen eine sehr ähnliche Zahl an benötigten Schritten.

5.4 Diskussion

Die in Abschnitt 5.1 vorgestellten Ergebnisse zeigen, dass das ERG-Verfahren durchaus in der Lage ist, gute Segmentierungsergebnisse zu berechnen. Unabhängig von der Größe des Bildes wurde bei einfachen horizontalen Schnitten stets das optimale Ergebnis erzielt. Auch bei einigen realen Bilddaten konnte das globale Maximum berechnet werden. Komplexere Beispiele wie eine gedrehte Gerade oder schwierige Objektformen führen allerdings zu einem Unterschied im berechneten Maximalfluss. Betrachtet man jedoch die Segmentierungen, so ist nur ein marginaler Unterschied zu der von CVX berechneten Lösung erkennbar. Das ERG-Verfahren arbeitet demnach approximativ, in einigen Fällen sogar optimal und weist in beiden Fällen gute Segmentierungsergebnisse auf.

Ein Vorteil der Erweiterung des ERG-Verfahrens ist die entstehende Unsicherheitsregion, in der unklar ist, welcher Menge der aktuelle Bildpunkt zugeordnet werden soll. Diese Region entsteht durch freie Pixel, denen im Laufe des Verfahrens kein eindeutiger Suchbaum zugewiesen werden konnte. Dadurch kann eine ähnlichere Lösung zu CVX bestimmt werden, da auch diese meist einen Bereich von nicht eindeutiger Zuordnung aufweist. Dies ist ein Vorteil, da auf dieser Basis weiterführend ein (subpixel-) genaueres Verfahren entwickelt werden kann.

Der große Vorteil dieses und anderer kontinuierlicher Verfahren ist die Isotropie. Insbesondere die synthetischen Testdaten zeigen die Richtungsabhängigkeit der diskreten Verfahren. Die Segmentierung eines Kreises führt bei den diskreten Verfahren mit einer 4er-Nachbarschaft zu der Kontur eines Quadrates. Der Übergang zu einem größeren Graphen kann bereits eine kreisähnliche Struktur segmentieren. Dagegen sind alle kontinuierlichen Verfahren nahezu richtungsunabhängig und berechnen den gewünschten Kreis. Ein größeres Nachbarschaftssystem kann ähnlich gute Ergebnisse erzielen, was sich auch in Abbildung 20 zeigt, jedoch führt es in einigen Fällen auch zum Verlust feiner Details. Bei größeren realen Bilddaten ist die Anisotropie der diskreten Verfahren weniger deutlich, dennoch ist die Präferenz bestimmter Richtungen erkennbar.

Im Rahmen einer Laufzeitanalyse ist jedoch das diskrete Verfahren mit einer 4er-Nachbarschaft das schnellste. Für kleinere Bilddaten ist die Implementierung mit CVX ebenfalls schnell, jedoch besitzt es für größere Bilder Speicherprobleme. Bei der Betrachtung der Skalierung der Laufzeit bei unterschiedlichen Bildgrößen weisen das ERG- und das diskrete Verfahren sehr ähnliche Tendenzen auf. Das ERG-Verfahren mit der Suche nach kürzesten Pfaden ist dabei unwesentlich langsamer als die diskrete Methode. Eine Übertragung des Quellcodes in C++ oder eine Implementierung auf der GPU könnte diesen Unterschied möglicherweise tilgen. Die Erweiterung des ERG-Verfahrens nach Boykov und Kolmogorov [11] ist bei größeren Bilddaten ebenfalls kaum verwendbar, da die Laufzeit stark ansteigt. Dies liegt vor allem an der MATLAB-Implementierung. Viele Schleifen und ein großer Speicheraufwand, der durch die Suchbäume zustande kommt, führen zu langsamen Pfadsuchen. Auch hier wäre eine Programmierung in C++ sinnvoll, um diese Nachteile auszugleichen. Im Zeitrahmen dieser Arbeit konnte eine solche Implementierung jedoch nicht erfolgen und stellt eine Möglichkeit zur Erweiterung dar.

			Laufzeit		
	Ansatz	Isotropie	kleine Bilder	große Bilder	Anzahl Schritte
CVX	Kontinuierlich	Ja	1	/	1
Graph-Cut(4)	Diskret	Nein	2	1	4
Graph-Cut(16)	Diskret	Eher ja	3	3	5
ERG	Kontinuierlich	Ja	4	2	2
ERG (BK)	Kontinuierlich	Ja	5	4	3

Tabelle 1. Vergleich der fünf untersuchten Verfahren hinsichtlich ihres Ansatzes, einer erkennbaren Isotropie, der Laufzeit für kleine und große Bilder sowie der Anzahl der benötigten Schritte. Bezüglich der Laufzeit und der Anzahl der Schritte bis zur Lösung des Problems wurde eine Rangfolge von 1 bis 5 erstellt. Eine 1 steht dabei für die schnellste Berechnungszeit beziehungsweise die wenigsten benötigten Iterationen. Für große Bilddaten ist CVX aufgrund von Speicherproblemen nicht verwendbar.

Im Vergleich zu dem Verfahren mit einer 16er-Nachbarschaft ist die gewöhnliche ERG-Methode bei größeren Bildern schneller. Auch die Zahl der benötigten Iterationen ist bei allen kontinuierlichen Methoden (auch bei der Erweiterung der ERG-Methode) geringer als bei den diskreten Ansätzen. Daher ist bei einer Implementierung der ERG-Methode mit Erweiterung in C++ ein Geschwindigkeitsanstieg zu erwarten.

Tabelle 1 zeigt schließlich alle fünf untersuchten Verfahren im Vergleich hinsichtlich ihres Ansatzes, einer erkennbaren Anisotropie, der Laufzeit für kleine und große Bilder sowie der Anzahl der benötigten Schritte. Auch hier wird deutlich, dass das ERG-Verfahren insbesondere für große Bilddaten die beste Wahl darstellt, da es zusätzlich zu seiner Anisotropie eine schnelle Laufzeit besitzt und nur wenig Iterationen zur Lösung des Problems benötigt.

Kapitel 6: Zusammenfassung und Ausblick

In dieser Arbeit wurden diskrete und kontinuierliche Graph-Cut-Probleme betrachtet. Viele der vorgestellten diskreten Verfahren haben das Ziel, die Laufzeit vorhandener Algorithmen zu verkürzen. Allerdings weisen viele Verfahren Diskretisierungsartefakte in der Kontur ihrer berechneten Segmentierung auf. Es wurde erläutert, inwiefern kontinuierliche Graph-Cut-Verfahren diese umgehen können. Eine alternative Betrachtung stellt beispielsweise die Definition der Kapazitäten auf den Bildpunkten statt auf den Kanten eines Graphen dar.

Das Hauptziel dieser Masterarbeit besteht in der Untersuchung und Erweiterung der Methode des erweiterten Residualgraphen, welche in einer Vorarbeit von Cremer [19] entwickelt wurde. In diesem Zusammenhang wurden zunächst einige Fehler in der von Cremer [19] erfolgten Implementierung korrigiert. Zusätzlich wurde der erweiterte Residualgraph insofern verändert, als dass die Quelle und Senke nicht weiter als vier Teilknoten dargestellt werden, sondern nur durch einen Knoten repräsentiert sind.

Ein zentraler Fokus der Analyse bestand in der Suche nach Pfaden innerhalb des erweiterten Residualgraphen. Da in jeder Iteration eine Pfadsuche vonnöten ist, wurde ihre bisherige Implementierung durch die vordefinierte MATLAB-Funktion `graphshortestpath` ersetzt, wodurch das Verfahren wesentlich beschleunigt wird. Um die Pfade nicht in jedem Schritt von neuem berechnen zu müssen, wurde das Verfahren nach Boykov und Kolmogorov [11] erweitert. Die Einführung zweier Suchbäume garantiert zwar nicht zwangsläufig die Berechnung der kürzesten Pfade, sorgt aber dafür, dass ein zuvor berechneter Pfad gespeichert und in der darauf folgenden Suche erneut verwendet werden kann. Da einige Bildpunkte innerhalb dieser Darstellung keinem Suchbaum eindeutig zugeordnet werden können, entsteht automatisch eine Unsicherheitsregion innerhalb der erzeugten Segmentierung, was eine Präzision des Ergebnisses zur Folge hat.

Ein weiteres Problem der von Cremer [19] erfolgten Implementierung bestand darin, dass das globale Maximum in den meisten Tests nicht erreicht wurde. Als Lösungsansatz wurde ein Quasizyklus definiert, der eine Verschiebung des Flusses zur Folge hat. Dies erlaubt es, zusätzlichen Fluss durch den Graphen fließen zu lassen. Somit kann das globale Maximum schließlich erreicht werden.

Die Ergebnisse aus Kapitel 5 zeigen, dass das ERG-Verfahren sehr gute Segmentierungsergebnisse berechnen kann. Auch wenn die Methode in manchen Fällen nur eine approximative Lösung bestimmt, ähneln die entstehenden Segmentierungen stark den von CVX ermittelten optimalen Ergebnissen. Im Gegensatz zu diskreten Ansätzen besitzt das ERG-Verfahren zudem keine erkennbare Richtungsabhängigkeit. Mit einer 4er-Nachbarschaft weist das untersuchte diskrete Verfahren zwar eine kürzere Laufzeit auf, benötigt jedoch eine größere Anzahl an Iterationen, um ein optimales Ergebnis zu berechnen als das ERG-Verfahren. Wird der Graph innerhalb des diskreten Verfahrens

auf 16 Nachbarn erweitert, so ist es bei größeren Bildern sogar langsamer als die ERG-Methode und löst das Problem mit einer bis zu sechsmal größeren Anzahl an Schritten.

Die Erweiterung des ERG-Verfahrens nach Boykov und Kolmogorov [11] berechnet ähnlich wie CVX eine Unsicherheitsregion auf Basis der freien Bildpunkte, wodurch die Genauigkeit der Methode verbessert werden kann. Die Vorteile dieser Erweiterung der Pfadsuche stehen einer noch hohen Laufzeit gegenüber. Da das Verfahren jedoch eine geringere Anzahl an Iterationen zur Lösung des Optimierungsproblems benötigt als die diskreten Ansätze, ist bei einer Implementierung in C++ ein entsprechender Geschwindigkeitsvorteil zu erwarten.

Obwohl die Lösung des kontinuierlichen Problems durch CVX stets optimale Ergebnisse berechnet, ist CVX für größere Bilddaten aufgrund von Speicherproblemen nicht verwendbar. Daher ist auch in diesem Fall das ERG-Verfahren die bessere Wahl.

Insgesamt gesehen stellt das ERG-Verfahren eine vielversprechende Alternative zu diskreten Graph-Cut-Verfahren dar. Es ist nicht nur in der Lage, gute Segmentierungen zu berechnen, es zeigt auch keine erkennbare Richtungsabhängigkeit. Diese Diskretisierungsartefakte können bei diskreten Verfahren durch eine Vergrößerung der Nachbarschaft umgangen werden. Dies geschieht jedoch auf Kosten von Details und der Laufzeit. Insbesondere in der Anzahl der benötigten Iterationen zum globalen Optimum weisen beide vorgestellten ERG-Varianten bessere Ergebnisse auf als die diskreten Verfahren. Durch die Erweiterung auf die Pfadsuche nach Boykov und Kolmogorov [11] kann schließlich die Genauigkeit des Verfahrens durch einen unsicheren Bereich verbessert werden.

6.1 Ausblick

Die Erweiterung des ERG-Verfahrens auf eine Pfadsuche mit zwei Suchbäumen liefert in den meisten Fällen gute Ergebnisse. Der gegebene Unsicherheitsbereich, ähnlich dem in der Lösung von CVX vorhandenen, ist ein Vorteil gegenüber der ursprünglichen Implementierung, wodurch die Segmentierung präziser wird. Weiterführend kann ein somit (subpixel-) genaues Verfahren entwickelt werden. Ein weiterer möglicher Schritt besteht darin, CVX über den entstandenen Unsicherheitsbereich des ERG-Verfahrens zu starten, um eine exakte und detaillierte Lösung zu berechnen.

Aufgrund der Programmierung in MATLAB ist die Erweiterung des ERG-Verfahrens mit zwei Suchbäumen jedoch langsam, was eine Umsetzung aller Funktionen in C++ nahelegt. Die Ergebnisse aus Kapitel 5 lassen eine Verkürzung der Laufzeit erwarten, da vergleichsweise wenig Pfade berechnet werden müssen, um das Problem zu lösen.

Eine zusätzliche Erweiterung besteht im Ausbau des erweiterten Residualgraphen auf dreidimensionale Bilder. Dazu müssten die Nachbarschaft und die Anzahl der Unterknoten für jeden Bildpunkt vergrößert werden.

Bislang arbeitet das ERG-Verfahren mit einem kantenbasierten Ansatz für Quelle und Senke, wodurch sie eigene Bildpunkte darstellen. Fortführend ist es möglich, die Methode auf einen regionenbasierten Ansatz zu erweitern. Dazu ist es nötig, den Ansatz in den kontinuierlichen Raum abzubilden, um ihn anschließend mithilfe des ERG-Verfahrens zu lösen.

Schließlich sind bislang keine Aussagen über eine garantierte Konvergenz des ERG-Verfahrens bewiesen. Ein geeigneter Beweis ist essentiell, um beurteilen zu können, wie gut das Verfahren das gegebene Problem wirklich lösen kann.

Literatur

[1] CVX Research: Software for Disciplined Convex Programming. http://cvxr.com/ (2012), accessed: 02.10.2016

[2] MEX-files guide from MathWorks. http://de.mathworks.com/help/matlab/call-mex-files-1.html?s_cid=wiki_mex_1 (2016), accessed: 29.09.2016

[3] Appleton, B., Talbot, H.: Globally minimal surfaces by continuous maximal flows. IEEE Transactions on Pattern Analysis and Machine Intelligence 28, 106–118 (2006)

[4] Arora, C., Banerjee, S., Kalra, P., Maheshwari, S.N.: An efficient graph cut algorithm for computer vision problems. In: European Conference Computer Vision. pp. 552–556 (2010)

[5] Bae, E., Yuan, J., Tai, X., Boykov, Y.: A fast continuous max-flow approach to non-convex multilabeling problems. Efficient Algorithms for Global Methods in Computer Vision 8293, 134–154 (2014)

[6] Bertsekas, D.P.: Network Optimization: Continuous and Discrete Models. Athena Scientific (1998)

[7] Bondy, J.A., Murty, U.S.R.: Graph theory with applications, vol. 290. Citeseer (1976)

[8] Boyd, S., Vandenberghe, L.: Convex optimization. Cambridge university press (2004)

[9] Boykov, Y., Funka-Lea, G.: Graph cuts and efficient n-d image segmentation. International Journal of Computer Vision 70(2), 109–131 (2006)

[10] Boykov, Y., Kolmogorov, V.: Computing geodesics and minimal surfaces via graph cuts. International Conference on Computer Vision 1, 26–33 (2003)

[11] Boykov, Y., Kolmogorov, V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. IEEE Transactions on PAMI 26(9), 1124–1137 (2004)

[12] Boykov, Y., Yuri, Y., Jolly, M.P.: Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In: Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on. vol. 1, pp. 105–112. IEEE (2001)

[13] Boykov, Y., Veksler, O., Zabih, R.: Markov random fields with efficient approximations. In: Computer vision and pattern recognition, 1998. Proceedings. 1998 IEEE computer society conference on. pp. 648–655. IEEE (1998)

[14] Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Transactions on pattern analysis and machine intelligence 23(11), 1222–1239 (2001)

[15] Bunke, H., Jiang, X.: Dreidimensionales Computersehen. Gewinnung und Analyse von Tiefenbildern (1997)

[16] Chambolle, A., Cremers, D., Pock, T.: A convex approach for computing minimal partitions (2008)

[17] Cormen, T.H.: Introduction to Algorithms. MIT Press (2001)

[18] Couprie, C., Grady, L., Talbot, H., Najman, L.: Combinatorial continuous maximum flow. SIAM Journal on Imaging Science 4(3), 905–930 (2011)

[19] Cremer, P.D.: Non-traditional maximal flow methods in image analysis. Essay 104, part iii of the mathematical tripos, University of Cambridge (2014)

[20] Demant, C., Streicher-Abel, B., Springhoff, A.: Industrielle Bildverarbeitung: Wie optische Qualitätskontrolle wirklich funktioniert. Springer-Verlag (2011)

[21] Dixit, N., Keriven, R., Paragios, N.: GPU-cuts: Combinatorial optimisation, graphic processing units and adaptive object extraction. Tech. Rep. 05-07, CERTIS, ENPC (2005)

[22] Edmonds, J., Karp, R.M.: Theoretical improvements in algorithmic efficiency for network flow problems. Journal of the ACM (JACM) 19(2), 248–264 (1972)

[23] Ford Jr, L.R., Fulkerson, D.R.: Flows in Networks. Princeton University Press (1962)

[24] Forster, O.: Analysis 3, Integralrechnung im $\mathbb{R}^n$ mit Anwendungen, 3. Au. Braunschweig: Vieweg (1984)

[25] Ghrist, R., Krishnan, S.: A topological max-flow-min-cut theorem. In: GlobalSIP. pp. 815–818 (2013)

[26] Goldberg, A.V., Tarjan, R.E.: A new approach to the maximum-flow problem. Journal of the ACM (JACM) 35(4), 921–940 (1988)

[27] Greig, D.M., Porteous, B.T., Seheult, A.H.: Exact maximum a posteriori estimation for binary images. Journal of the Royal Statistical Society. Series B (Methodological) pp. 271–279 (1989)

[28] Handels, H.: Medizinische Bildverarbeitung: Bildanalyse, Mustererkennung und Visualisierung für die computergestützte ärztliche Diagnostik und Therapie. Springer-Verlag (2009)

[29] Hofmann-Wellenhof, B., Moritz, H.: Physical geodesy. Springer Science & Business Media (2006)

[30] Hussein, M., Varshney, A., Davis, L.: On implementing graph cuts on CUDA. In: First Workshop on General Purpose Processing on Graphics Processing Units (2007)

[31] Juan, O., Boykov, Y.: Capacity scaling for graph cuts in vision. In: International Conference on Computer Vision. pp. 1–8 (2007)

[32] Klodt, M., Schoenemann, T., Kolev, K., Schikora, M., Cremers, D.: An experimental comparison of discrete and continuous shape optimization methods. In: European Conference on Computer Vision. pp. 332–345. Springer (2008)

[33] Kolmogorov, V., Zabih, R.: What energy functions can be minimized via graph cuts? IEEE Transactions on Pattern Analysis and Machine Intelligence 26(2), 147–159 (2004)

[34] Kolmogorov, V., Boykov, Y.: What metrics can be approximated by geo-cuts, or global optimization of length/area and flux? 1, 564–571 (2005)

[35] Kolmogorov, V., Zabin, R.: What energy functions can be minimized via graph cuts? IEEE Transactions on Pattern Analysis and Machine Intelligence 26(2), 147–159 (2004)

[36] Komodakis, N., Tziritas, G.: Approximate labeling via graph cuts based on linear programming. Patt. Anal. Mach. Intell. 29(8), 1436–1453 (2007)

[37] Korte, B., Vygen, J., Korte, B., Vygen, J.: Combinatorial optimization, vol. 2. Springer (2012)

[38] Krishnan, S.: Flow-cut dualities for sheaves on graphs. arXiv preprint arXiv:1409.6712 (2014)

[39] Lawler, E.L.: Combinatorial optimization: networks and matroids. Courier Corporation (2001)

[40] Lellmann, J., Lellmann, B., Widmann, F., Schnörr, C.: Discrete and continuous models for partitioning problems. International journal of computer vision 104(3), 241–269 (2013)

[41] Munkres, J.R.: Elements of algebraic topology, vol. 2. Addison-Wesley Menlo Park (1984)

[42] Owens, J.D., Houston, M., Luebke, D., Green, S., Stone, J.E., Phillips, J.C.: Gpu computing. Proceedings of the IEEE 96(5), 879–899 (2008)

[43] Papadimitriou, C.H., Steiglitz, K.: Combinatorial optimization: algorithms and complexity. Courier Corporation (1982)

[44] Rother, C., Kolmogorov, C., Blake: Grabcut: Interactive foreground extraction using iterated graph cuts. In: ACM Trans. Graphics. vol. 23, pp. 309–314 (2004)

[45] Rüdiger, K.: Medizintechnik: Verfahren–Systeme–Informationsverarbeitung. 2 (2001)

[46] Schmidt, F.R., Töppe, E., Cremers, D.: Efficient planar graph cuts with applications in computer vision. Proceedings of the IEEE Computer Vision and Pattern Recognition 1, 351–356 (2009)

[47] Sethian, J.A.: Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science, vol. 3. Cambridge university press (1999)

[48] Strang, G.: Maximum flows and minimum cuts in the plane. Advances in Mechanics and Mathematics 3, 1–10 (2008)

[49] Strang, G.: Maximal flow through a domain. Mathematical Programming 26(2), 123–143 (1983)

[50] Vineet, V., Narayanan, P.J.: CUDA cuts: Fast graph cuts on the GPU. In: Comp. Vis. Patt. Recogn. pp. 1–8 (2008)

[51] Wallis, W.D.: A beginner's guide to graph theory. Springer Science & Business Media (2010)

[52] Woods, R.P.: Handbook of medical image processing and analysis. Within-Modality Registration Using Intensity-Based Cost Functions 33, 529–536 (2000)

[53] Yuan, J., Bae, E., Tai, X.C.: A study on continuous max-flow and min-cut approaches. In: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on. pp. 2217–2224. IEEE (2010)

[54] Yuan, J., Bae, E., Tai, X.C., Boykov, Y.: A continuous max-flow approach to potts model. In: European Conference on Computer Vision. pp. 379–392. Springer (2010)

Kapitel A: Anhang

A.1 Algorithmus von Boykov und Kolmogorov [11]

Algorithmus 12 Algorithmus von Boykov und Kolmogorov

Initalisiere $S = \{s\}, T = \{t\}, A = \{s, t\}$: Menge der aktiven Knoten, $O = \emptyset$: Menge der Waisen
Solange „wahr“ ▷ *Wachstum* von S oder T
    Solange $A \neq \emptyset$
        Nimm aktiven Knoten $p \in A$
        Für alle Nachbarn q mit $c(p, q) > 0$
            Falls $tree(q) = \emptyset$
                $tree(q) = tree(p), parent(q) = p, A = A \cup \{q\}$
            Falls $tree(q) \neq \emptyset$ und $tree(q) \neq tree(p)$
                Gib $P = PATH_{s \to t}$ zurück
    Gib $P = \emptyset$ zurück

    Finde größtmögliche Flusserhöhung Δ auf P und aktualisiere G_f ▷ *Augmentiere* auf P
    Für Kanten $(p, q) \in P$, die gesättigt werden
        Falls $tree(p) = tree(q) = S$
            $PAparent(q) = \emptyset, O = O \cup \{q\}$
        Falls $tree(p) = tree(q) = T$
            $parent(p) = \emptyset, O = O \cup \{p\}$

    Solange $O \neq \emptyset$ ▷ *Adoptiere* Waisen
        Nimm Waise $p \in O$ und entferne p aus O
        Für alle Nachbarn q von p
            Falls $tree(p) = tree(q), c(q, p) > 0$ und Ursprung von q ist s bzw. t
                $parent(p) = q$
        Für alle Nachbarn q von p
            Falls Nachbar q mit $tree(q) = tree(p)$
                Falls $c(q, p) > 0$
                    $A = A \cup \{q\}$
                Falls $parent(q) = p$
                    $O = O \cup \{q\}, parent(q) = \emptyset$
            $tree(p) = \emptyset, A = A - \{q\}$

A.2 Beweis von Satz 2 nach [19]

Beweis.

" $\Rightarrow$ ": Sei P_f ein Pfad im erweiterten Residualgraphen G_f. Dieser wird wie folgt konstruiert: Für jeden inneren Knoten v auf P betrachte den Vorgänger x und Nachfolger y. Somit werden die Knoten x_i, v_j, v_k und y_l zu P_f hinzugefügt, wobei

$$\begin{aligned} &i = 1, j = 3, && \text{wenn } x \text{ links neben } v \text{ liegt,} \\ &i = 3, j = 1, && \text{wenn } x \text{ rechts neben } v \text{ liegt,} \\ &i = 2, j = 4, && \text{wenn } x \text{ oberhalb von } v \text{ liegt,} \\ &i = 4, j = 2, && \text{wenn } x \text{ unterhalb von } v \text{ liegt.} \end{aligned} \tag{67}$$

Die gleiche Ordnung folgt für die Knoten v_k und y_l. Schließlich werden s_1 und t_1 an das Ende beziehungsweise den Anfang des Pfades hinzugefügt. Da P zulässig ist und die Kanten in G_f nur hinzugefügt werden, wenn sie zulässige Richtungen beinhalten, ist P_f ein augmentierender Pfad.

" $\Leftarrow$ ": Für die Rückrichtung wird Lemma 4.1 aus [19] verwendet:

Lemma 1. Seien $(v_i, v_j), (v_j, v_k) \in E(G_f)$ Kanten gleicher Elternknoten mit dem Elternknoten v und $v_i \neq v_k$. Dann gilt $(v_i, v_k) \in E(G_f)$.

Der Beweis ist in [19] zu finden. Somit lässt sich ein neuer gültiger Weg konstruieren, indem zwei aufeinanderfolgende Kanten der gleichen Elternknoten durch eine einzelne Kante ersetzt wird bis keine aufeinanderfolgenden Kanten gleicher Elternknoten mehr vorhanden sind.
Beinhaltet P_f zwei aufeinanderfolgende Kanten zwischen unterschiedlichen Elternknoten, so haben diese die Form $(v_i, w_j), (w_j, v_i)$ aufgrund der Konstruktion von G_f. Dies widerspricht jedoch der Definition eines Pfades, wonach jeder Knoten nur ein einziges Mal auftauchen kann (siehe Definition 4).
Demnach lässt sich folgern, dass P_f aus alternierenden Kanten zwischen gleichen und zwischen unterschiedlichen Elternknoten besteht. Cremer [19] behauptet nun, dass der Weg P, der die Eltern von Kanten zwischen unterschiedlichen Elternknoten beinhaltet, ein zulässiger s-t-Weg in G ist. Da der Weg offensichtlich ein s-t-Weg ist, bleibt die Zulässigkeit der inneren Knoten zu prüfen.
Sei $x_{d_{i-1}}, v_{d_i}, v_{d_{i+1}}, y_{d_{i+2}}$ eine Teilfolge des Pfades, welche mit einer Kante zwischen unterschiedlichen Elternknoten startet. Taucht der Knoten v nur ein einziges Mal in dem Weg P auf, garantiert die Existenz der Kante zwischen gleichen Elternknoten $(v_{d_i}, v_{d_{i+1}})$ in G_f, dass die Kapazitätsbedingung (51) erfüllt ist. Ist v zweimal in P zu finden, so lassen sich alle Unterknoten in P_f finden, da P_f ein Pfad ist und kein Unterknoten zweifach auftauchen darf. Dies zeigt ebenfalls, dass kein v mehr als zweimal auftauchen kann. Somit sind alle Richtungen, in denen v betreten wird und links im Weg P auftauchen, eindeutig. Die Paare ein- und austretenden

Flusses in G_f werden im Folgenden mit $(1, 2)$ und $(3, 4)$ bezeichnet. Es bleibt nun zu zeigen, dass

$$(f_1 + \epsilon)^2 + (f_2 - \epsilon)^2 + (f_3 + \epsilon)^2 + (f_4 - \epsilon)^2 \leq c_v^2 \tag{68}$$

für ein $\epsilon > 0$. Wir wissen, dass

$$\exists \epsilon_1 > 0 : \quad (f_1 + \epsilon)^2 + (f_2 - \epsilon)^2 + f_3^2 + f_4^2 \leq c_v^2 \tag{69}$$

$$\exists \epsilon_2 > 0 : \quad f_1^2 + f_2^2 + (f_3 + \epsilon)^2 + (f_4 - \epsilon)^2 \leq c_v^2 \tag{70}$$

$$f_1^2 + f_2^2 + f_3^2 + f_4^2 \leq c_v^2. \tag{71}$$

Es existieren zwei Fälle. Wenn Ungleichung (71) eine Gleichheit aufweist, garantiert nach [19] die Stetigkeit eine Existenz von $\epsilon > 0$, so dass die Ungleichung (68) erfüllt ist. Andernfalls erhalten wir (68), wenn die Summe von (69) und (70) mit $\epsilon = \epsilon_1 = \epsilon_2$ von (71) subtrahiert wird.
Somit existiert für alle Knoten v in dem Weg P ein $\epsilon_v > 0$, so dass die Kapazitätsbedingung (51) erfüllt ist. Da der Weg nur eine endliche Länge besitzt, existiert ein $\epsilon > 0$, so dass (51) für alle inneren Knoten erfüllt ist. Daher ist der konstruierte Weg in G zulässig.

□